

NASA CR-

147453

TRW NOTE NO. 74-FMT-937

JSC/TRW TASK 531

LEVEL II MDAS PROTOTYPE MONITOR
PROGRAM DOCUMENT
(PART II)

14 June 1974

Prepared for
Mission Planning and Analysis Division
National Aeronautics and Space Administration
Johnson Space Center
Houston, Texas

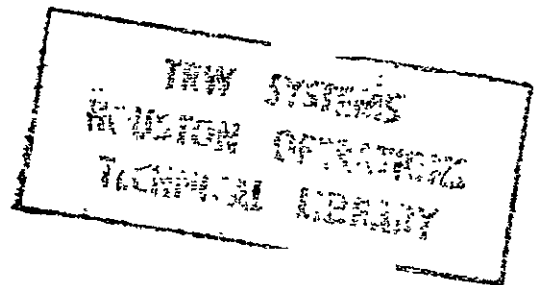
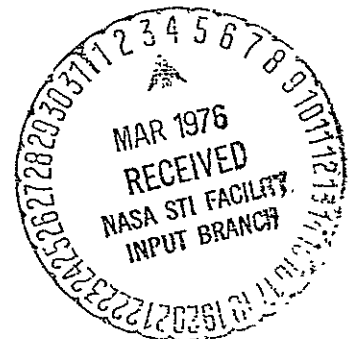
NAS 9-13834

Prepared by
Systems Evaluation Department

(NASA-CR-147453) LEVEL 2 MDAS PROTOTYPE
MONITOR PROGRAM DOCUMENT, PART 2. (TRW
Systems Group) 289 p HC \$9.25 CSCI 09B

N76-18825

Unclas
18691
G3/61



TRW NOTE NO. 74-FMT-937

JSC/TRW TASK 531


LEVEL II MDAS PROTOTYPE MONITOR
PROGRAM DOCUMENT
(PART II)


14 June 1974

Prepared for
Mission Planning and Analysis Division
National Aeronautics and Space Administration
Johnson Space Center
Houston, Texas

NAS 9-13834

Prepared by
Systems Evaluation Department


E. L. Ellisor, Jr., Task Manager
JSC/TRW Task 531


R. K. Petersburg, Manager
Systems Evaluation Department

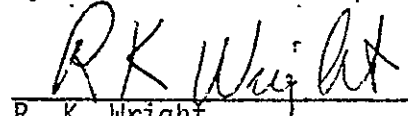

R. K. Wright
Assistant Project Manager
VMMPS
Mission Trajectory Control Program

Table of Contents

<u>Date Delivered</u>		<u>Page</u>
4/30	1. Resident	1.1-1
4/30	2. Monitor Boot Logic	2.1-1
	3. User Communications	
5/31	3.1 MDBC DI	3.1-1
5/31	3.2 MDBC I2	3.2-1
5/31	3.3 MDC DAT	3.3-1
5/31	3.4 MD CONV	3.4-1
5/31	3.5 MD PCK	3.5-1
4/30	3.6 MD PRMH	3.6-1
4/30	3.7 MD PRMI	3.7-1
4/30	3.8 MD PRMR	3.8-1
5/31	3.9 MD PRMT	3.9-1
5/31	3.10 MD SCAN	3.10-1
5/31	3.11 MD SQZB	3.11-1
	4. Storage Monitor	
5/31	4.1 MD ELET	4.1-1
5/31	4.2 MD ENTR	4.2-1
5/31	4.3 MD FIND	4.3-1
5/31	4.4 MD GET	4.4-1
5/31	4.5 MD PACK	4.5-1
5/31	4.6 MD PUT	4.6-1
5/31	4.7 MD RADI	4.7-1
5/31	4.8 MD ROLL	4.8-1
	5. Execution Controller	
5/31	5.1 MD ALOC	5.1-1
4/30	5.2 MD CMTG	5.2-1
4/30	5.3 MD CMTV	5.3-1
5/31	5.4 MD MERG	5.4-1
5/31	5.5 MD PRT	5.5-1
6/14	5.6 MD SMON	5.6-1

Table of Contents (Continued)

<u>Date Delivered</u>		<u>Page</u>
	6. Command Table Editor	
4/30	6.1 MDCMNT	6.1-1
4/30	6.2 MDCMT	6.2-1
4/30	6.3 MDCMTL	6.3-1
4/30	6.4 MDCMTS	6.4-1
4/30	6.5 MDVCMD	6.5-1
	7. Control Table Editor	
5/31	7.1 MDALST	7.1-1
5/31	7.2 MDCNT	7.2-1
5/31	7.3 MDCNTE	7.3-1
5/31	7.4 MDCNTS	7.4-1
5/31	7.5 MDCONT	7.5-1
5/31	7.6 MDDEFN	7.6-1
5/31	7.7 MDEDIT	7.7-1
5/31	7.8 MDSPEC	7.8-1
	8. Utility Routines	
5/31	8.1 MDALCT	8.1-1
5/31	8.2 MDCTPK	8.2-1
5/31	8.3 MDGETC	8.3-1
5/31	8.4 MDLIST	8.4-1
5/31	8.5 MDLKUP	8.5-1
4/30	8.6 MDLSTH	8.6-1
4/30	8.7 MDLSTI	8.7-1
4/30	8.8 MDLSTO	8.8-1
4/30	8.9 MDLSTR	8.9-1
5/31	8.10 MDPUTC	8.10-1
5/31	8.11 MDQUIT	8.11-1
4/30	8.12 MDSMTW	8.12-1
5/31	8.13 MDSPLT	8.13-1
4/30	8.14 MDTOC	8.14-1
6/14	8.15 MDUTIL	8.15-1
6/14	8.16 SEARCH	8.16-1
6/14	8.17 SORT2	8.17-1



Table of Contents (Continued)

<u>Date Delivered</u>		<u>Page</u>
	9. Library Maintenance	
5/31	9.1 DCTMOD	9.1-1
4/30	9.2 MDADDR	9.2-1
5/31	9.3 MDBULD	9.3-1
5/31	9.4 MDGENR	9.4-1
	10. IMS Interface	
5/31	10.1 MDIMS	10.1-1
	11. Access Control	
5/31	11.1 MDLOGO	11.1-1
6/14	11.2 MDELAC	11.2-1
5/31	Appendix A - Cross Reference	A-1
6/14	Appendix B - Common Blocks	B-1

LEVEL II MDAS PROTOTYPE MONITOR
PROGRAM DOCUMENT
(PART II)

1. Introduction

The purpose of this document is to provide detailed information about each subroutine contained in the monitor. This information is provided at a level such that programmers may become familiar with the design and techniques used to implement each component described in Part I of this document. However, flow charts are not provided for the extremely simple routines. This document is not intended to describe the user interface and should not be used as such. For this type of information see the "User's Guide for the Level II Mission Design and Analysis Subsystem (MDAS) Prototype"

In order to facilitate the reading of the flow charts contained herein the following convention was adopted for off page connectors. Each connector will contain a letter and a number, separated by a slash. The letter uniquely determines a destination and the number(s) refers to the page of the flow chart where the referenced connector is located or from where this connector is referenced. For example, ...  (A/2) would indicate the program flow is continued at connector A on page 2 of the flow chart. On page 2 we might find (A/1,3)  ... which indicates that this portion of the routine may be entered from either page 1 or page 3 of the flow.

This document describes the routines contained in the baseline program as delivered to JSC/MAB on 28 December 1973 on the CSC INFONET system.

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDAS Resident

The resident is the only program logic which is continuously present in memory during an MDAS session. It contains minimal logic to load and execute, and if required, scan the required processors and reload and return control to the submonitor.

Processing

Inputs All communications with the resident are via the intramonitor communications area. There are three classes of information used for processor loading 1) the library catalog, 2) processor calling sequence data, and 3) scan control values, all of which are described in Appendix C.

In addition to these inputs, the resident accesses the library files (Part I, Section 6) and reads the contents into the processor swap area as required.

Method The resident invokes the INFONET basic file services package LRS (Logical Record Services) which is system resident and thus external to the MDAS region to perform all input/output functions. A major portion of the resident code is directed toward manipulating the LRS file control blocks (UCB's, unsanctioned control blocks, and OCB's, operations control blocks). All communications with LRS is via these control blocks which must contain such parameters as file name and version, record length, buffer origin and length, access codes, file organization keys, etc.

The resident uses the catalog entry number to access the file name/version and origin and length data for the processor to be executed. This information is stored in the LRS control blocks, the file opened, and the input of the instruction bank (record) of the processor started. While the record is being loaded by LRS, the resident constructs the processor calling sequence if any. (The submonitor, loaded in a similar fashion, has no calling sequence).

The absolute addresses of the calling arguments are computed by adding the origin address of BLANK COMMON to the relative common address of the arguments as returned by the submonitor. A transfer instruction to the post processor execution logic is placed after the last argument address since this is the point of return following processor execution.

After the instruction bank input is complete, the data bank (record) is loaded and control is transferred to the processor. This sequence is repeated, alternating between processor loading and submonitor loading unless the submonitor indicates that a parametric scan of processor inputs is to be performed.

Scans are performed with a single processor loading as follows:

- 1) The processor inputs will have been defined by the submonitor such that the scan centroid point will be evaluated on the initial execution. The associated data box file will also have been opened and identification records output to it.
- 2) The resident will copy the complete summary table (name, units, and summary vectors) to the data box file. Adjustment of the input parameters to the (X_1, Y_1) point will then be accomplished and the processor re-executed.
- 3) Each subsequent return from the processor will be followed by copying the summary vector to the data box file and the scan parameters incremented such that the sequence $(X_2, Y_1), (X_3, Y_1) \dots (X_n, Y_1), (X_1, Y_2) \dots (X_n, Y_m)$ is completed.
- 4) In order that all output quantities of the processor will be left with values corresponding to the centroid point, the resident deactivates the scan, closes the data box file, resets the scan parameters to the centroid values, and executes the processor a final time.

Normal MDAS termination is accomplished by the transfer of control to the system directly from the submonitor.

Outputs Aside from fatal error messages the only outputs from the resident are the summary table contents of data box files and the deactivating of the scan activation flag.

USAGE

THE MDAS RESIDENT IS ENTERED FROM THE BOOT LOGIC BY THE FOLLOWING SEQUENCE OF ASSEMBLY LANGUAGE OPERATIONS

STORE ASCII NAME OF FUNCTION TO BE LOADED IN PRONAM
LOAD REGISTER B10 WITH PROTAB OFF SET MINUS ONE OF THE
FUNCTION I-BANK LENGTH AND ADDRESS WORD
JUMP TO LABEL INTIIN IN THE RESIDENT

EXTERNAL REFERENCES

ECLOS\$ TO CLOSE FILES
ECTSOS TO OUTPUT TO TERMINAL
ELRSRS TO READ LOGICAL RECORDS
ELRSWS TO WRITE LOGICAL RECORDS
EOPENS TO OPEN FILES
EROOLS TO TERMINATE EXECUTION
EWAITS TO WAIT FOR ASYNCHRONOUS READ COMPLETION
MDADDR FOR LIBRARY MAINTENANCE
MDSMON TO PERFORM MONITOR FUNCTIONS
SUBSYSTEM PROCESSORS AS REQUESTED BY USER

DIAGNOSTICS

D B WRITE ERROR, ID IN A1
THE CURRENTLY ACTIVE DATA BOX FILE CAN NOT BE COMPLETED
BY WRITING OF SUMMARY VECTOR. THE SYSTEM RETURN CODE IS
CONTAINED IN REGISTER A1.
PROC READ ERROR, ID IN A1
THE REQUESTED PROCESSOR LOAD MODULE FILE CAN NOT BE
READ. THE SYSTEM RETURN CODE IS CONTAINED IN REGISTER
A1.

EXTERNAL STORAGE

THE LOAD MODULE FILES OF THE SUBMONITOR DESIGNATED PROCESSORS
AND THE SUBMONITOR ARE ACCESSED FOR READ AND CLOSED.
THE DATA BOX FILES OPENED AND INITIALIZED BY THE SUBMONITOR
ARE COMPLETED AND CLOSED.

BLANK COMMON

VARB I/O

ARGADD I
CENTX I
CENTY I
DBOCB I/O
DBUCB I
DBSVLN I
INITX I
INITY I
NUMARG I/O
N2X I
N2Y I
PRONAM I
PRONUM I
PROTAB I
SCANF I/O

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

VARB I/O

XINC I

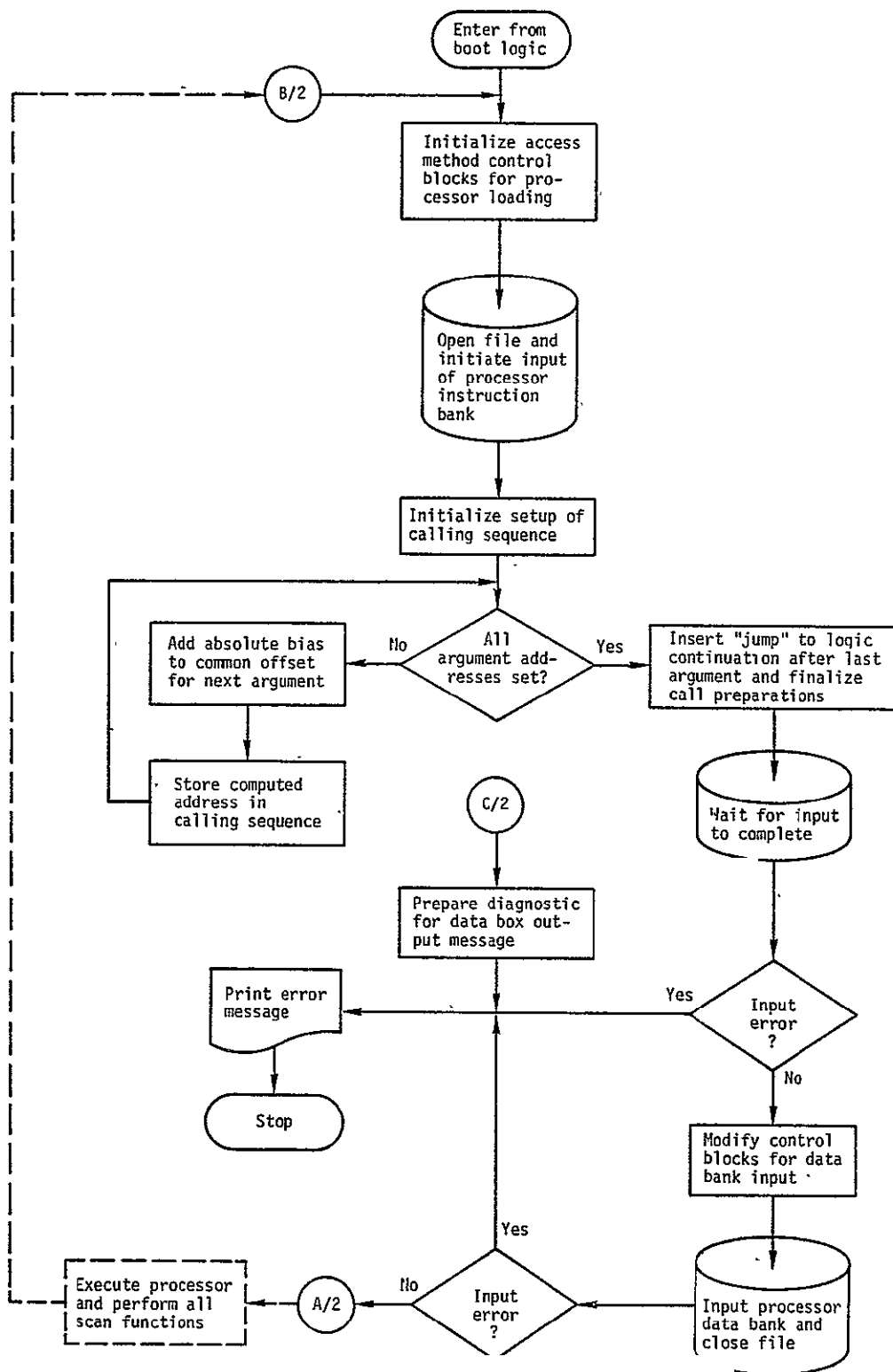
YINC I

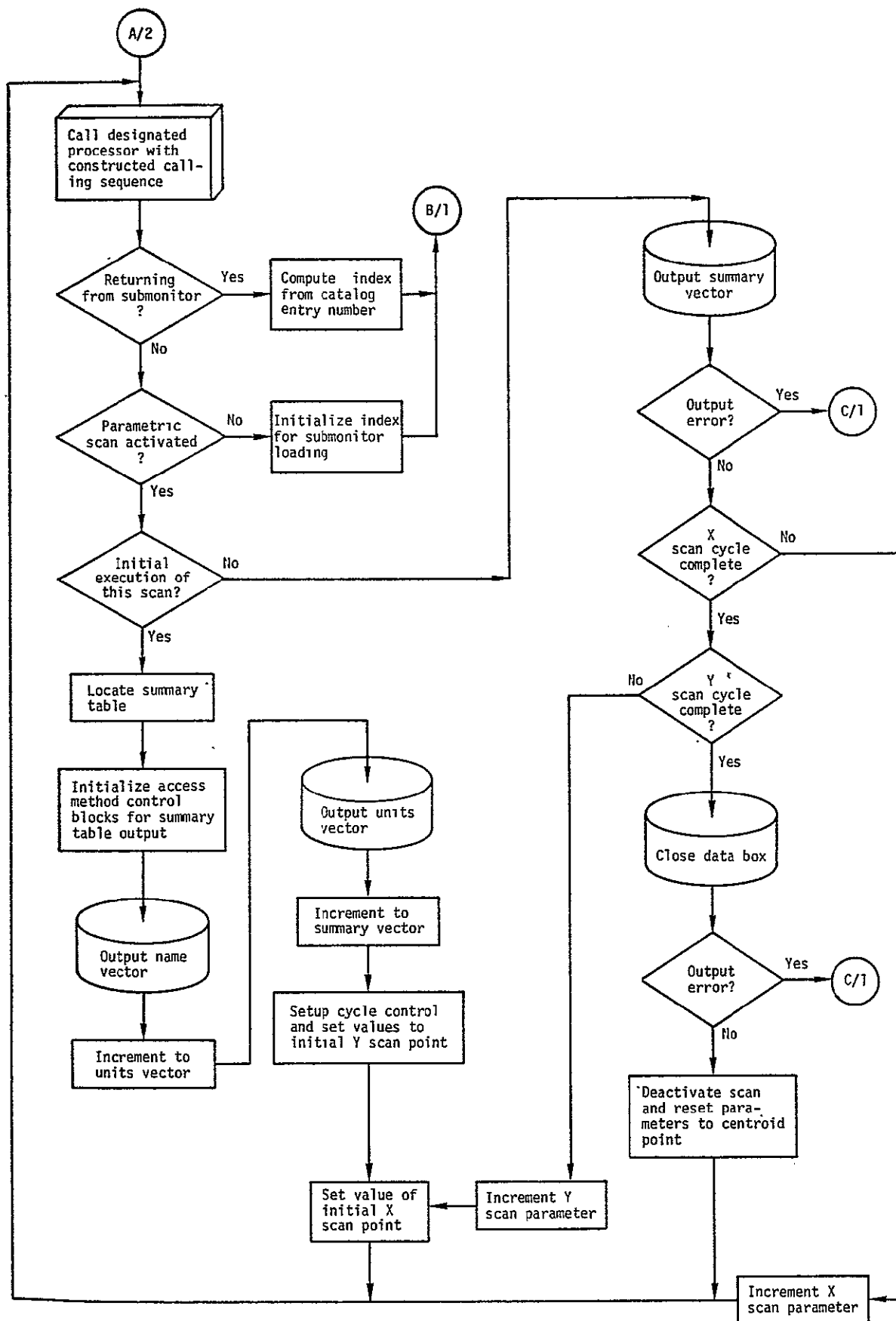
WORKX I/O

WORKY I/O

LOCAL COMMON

NONE





Boot Logic (MDAS)

The purpose of the boot logic (entry point MDAS) is to perform sufficient, one time initialization to bring the subsystem up for a user session. The basic tasks involved are the input of the library catalog from mass storage into the intramonitor communications area and the initializing of the resident/submonitor communications link such that transfer of control to the resident will cause it to load and execute the appropriate monitor function,

Processing

Inputs The only input to the boot logic is the library catalog which resides in mass storage file MDTABL.MD. This catalog and its associated header record are loaded into the intramonitor communications area (see Appendix C for locations).

Method The boot logic invokes the INFONET basic file services package LRS (Logical Record Services) which is system resident and thus external to the MDAS region to perform all input/output functions. A major portion of the code is directed toward manipulating the LRS file control blocks (UCB's, unsanctioned control blocks, and OCB's, operations control blocks). All communications with LRS is via these control blocks which must contain such parameters as file name and version, record length, buffer origin and length, access codes, file organization keys, etc.

The catalog file is opened and the three word header record input. These words contain size and pointer data to the remaining portions of the catalog. (Table 2-1 depicts the general structure, content and definitions of the catalog file MDTABL.MD.). From the header information the total space required for reading the catalog into memory is computed and tested against the size available.

The catalog used on the previous MDAS execution, referred to as the old catalog record, is input followed by any new processor entries into the catalog generated by the maintenance program MDGENR. As indicated in Appendix C, the remaining portion of the intramonitor communications area is reserved for an ephemeris data buffer and the storage monitor table (SMT);

thus the boot logic sets the origin of the buffer to the next available address following the catalog and the origin of the SMT following the fixed length buffer.

After the catalog input and associated allocations are accomplished, the boot logic primes the communications link to the resident to cause loading of the appropriate monitor function. There are three possibilities.

If no changes have been made to the library and catalog the submonitor (MDSMON) will be queued for loading. The necessary control block data is found in words two and three of the first catalog entry.

If MDGENR has modified the library and catalog the maintenance processor (MDADDR) must be loaded to reorganize the catalog and build or modify the default control tables for the affected processors. The necessary control block data is found in words four and five of the first catalog entry provided that MDADDR itself has not been modified.

The third possibility for monitor function loading is the invoking of a new version of MDADDR. This is detected by the presence of a non-zero value in the third word of the catalog header record. In such cases this value is the record number of the new catalog entry record corresponding to the revised MDADDR. To queue this function the associated control block data in words two and three of the new entry are referenced.

Before transferring control to the resident, the adequacy of the swap area region sizes is verified to insure proper loading of the queued function by the resident.

Outputs The appropriate portions of the intramonitor communications area are initialized as described and a monitor function is queued for loading by the resident. Several diagnostics related to detection of fatal errors may be output by the boot logic.

USAGE

ENTRY MDAS
EXECUTE THE GPS COMMAND !MDAS

EXTERNAL REFERENCES

MDAS RESIDENT

DIAGNOSTICS

CATALOG SIZE TO LARGE FOR PROTAB
THE MDAS LIBRARY CATALOG FILE WILL NOT FIT IN THE
INTRAMONITOR COMMUNICATIONS AREA AS PRESENTLY
CONFIGURED. EDIT MDAS-PNC TO REVISE THE VALUE OF CRES
APPROPRIATELY AND REASSEMBLE AND LINK MDAS.
EXTENT OF (.....) IS TOO LARGE FOR CURRENT MDAS
CONFIGURATION (.....).
THE INDICATED LOAD MODULE (MDSMON OR MDADDR) REQUIRES
A SWAP AREA LARGER THAN THE ALLOCATED REGIONS.
DETERMINE THE REQUIREMENTS, EDIT MDAS-PNC TO REVISE THE
VALUES OF IRES AND DRES APPROPRIATELY AND REASSEMBLE AND
LINK MDAS.
INIT READ ERROR, ID IN A1
THE BOOT LOGIC WAS UNABLE TO READ THE LIBRARY CATALOG
FILE MDTABL.MD. THE SYSTEM RETURN CODE IS CONTAINED IN
REGISTER A1.

EXTERNAL STORAGE

THE LIBRARY CATALOG FILE MDTABL.MD IS OPENED, ACCESSED FOR
READ AND CLOSED.

BLANK COMMON

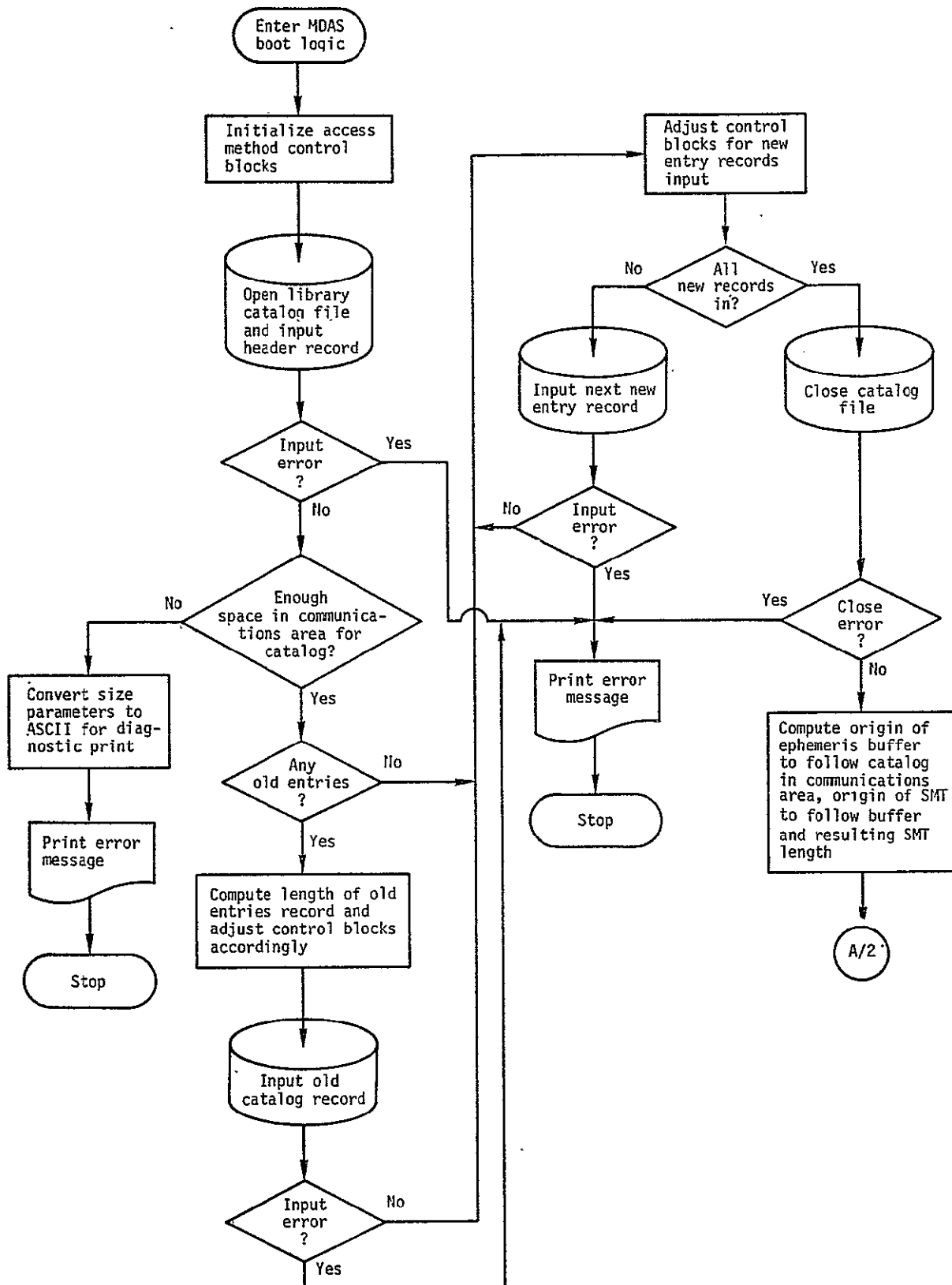
IN ADDITION TO THE FOLLOWING VARIABLES, THE BOOT LOGIC ITSELF
IS CONTAINED IN BLANK COMMON. IT CAUSES ITS OWN DESTRUCTION
TO OCCURE WHEN THE FIRST LOAD OF THE MONITOR IS ACCOMPLISHED
BY THE RESIDENT.

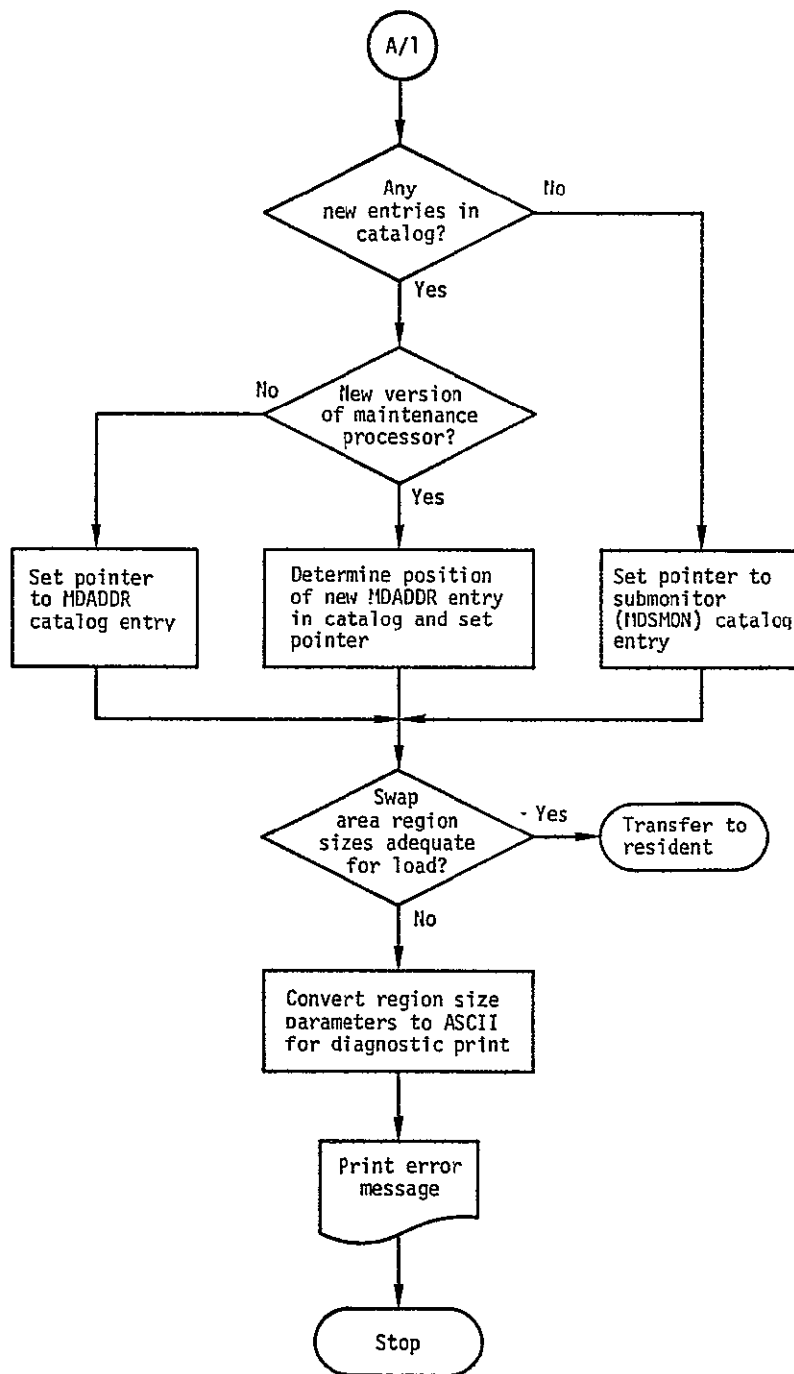
VARB I/O

DBSTRT	0
EPHLEN	1
EPSTRT	0
FIXCOM	1
NTRY	0
PRONAM	0
PROTAB	0
PTABKY	0

LOCAL COMMON

NONE





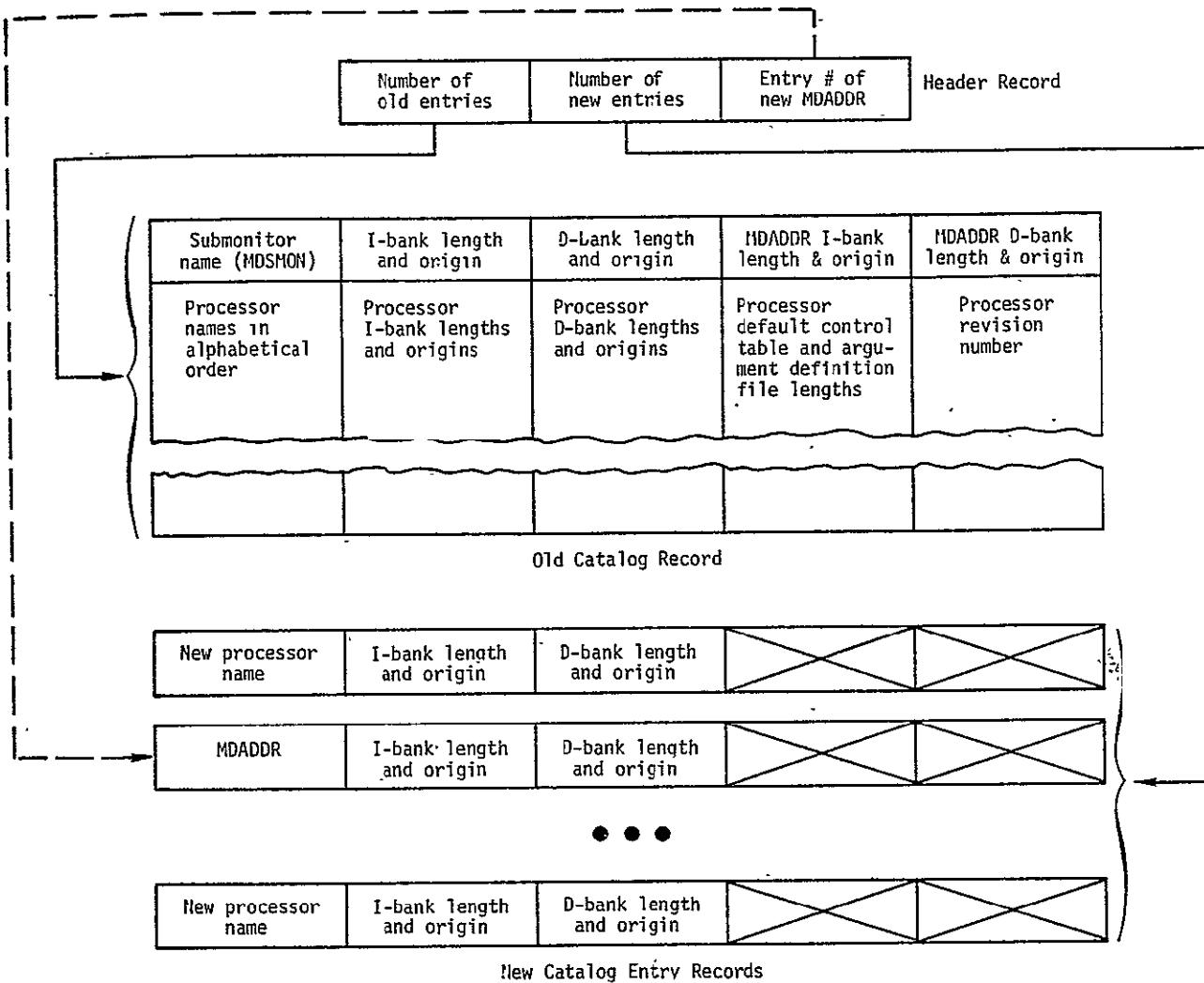


Table 2-1 Organization of the Library Catalog File MDTABL.MD

MDBCDI - User Communication

MDBCDI converts a BCD number to a binary integer and entry point MDIBCD converts a binary integer to a BCD number.

Method

Input: The inputs to MDBCDI are BCD numbers, one character per word to be converted, and the number of characters to convert. The inputs to entry point MDIBCD are the binary integer and the number of words available in the output array.

Processing: Routine MDBCDI converts BCD numbers to a binary integer. The BCD characters are input one character per word. MDSQZB is called to remove blanks from input string. MDBCDI checks each word for a digit (0 - 9). If the word does not contain a digit, an error flag is set to the negative of the subscript word number in error and the routine returns. The input number of words, i.e., BCD numbers, in the input array are converted to a binary output integer and the routine returns. MDBCDI performs the same function as MDBCD2 except MDBCD2 does not allow intervening blanks in the input character string.

Entry point MDIBCD converts a binary integer to a BCD number which is returned one character per word in the output array. The number of words left unfilled is output or if the number of digits exceed the available words, the negative of that number is output.

Output: The output from MDBCDI are the binary integer and an error flag if a digit or blank was not input in the BCD array. The outputs from MDIBCD are BCD numbers one character per word and the number of unfilled words or the negative of the extra words needed.

USAGE

ENTRY MDBC DI
CALL MDBC DI (NCOL,INT,N)

ARGMT	I/O	TYPE	DIM	DEFINITION
NCOL	I	I	N	ARRAY CONTAINING BCD NUMBER, ONE CHARACTER PER WORD
INT	O	I	1	BINARY INTEGER
N	I/O	I	1	NUMBER OF BCD WORDS INPUT IF A NON-DIGIT WAS INPUT IN ARRAY NCOL, ON OUTPUT N WILL BE A NEGATIVE VALUE WITH THE MAGNITUDE BEING THE SUBSCRIPT OF THE INCORRECT WORD

ENTRY MDIBCD
CALL MDIBCD (NCOL,INT,N)

ARGMT	I/O	TYPE	DIM	DEFINITION
NCOL	O	I	N	ARRAY CONTAINING BCD NUMBER, ONE CHARACTER PER WORD
INT	I	I	1	BINARY INTEGER
N	I/O	I	1	NUMBER OF WORDS AVAILABLE IN NCOL ON OUTPUT IT IS THE NUMBER OF UNFILLED WORDS OR THE NEGATIVE OF THE NUMBER OF EXTRA WORDS NEEDED IF NCOL IS OVERFLOWED

EXTERNAL REFERENCES
MDSQZB

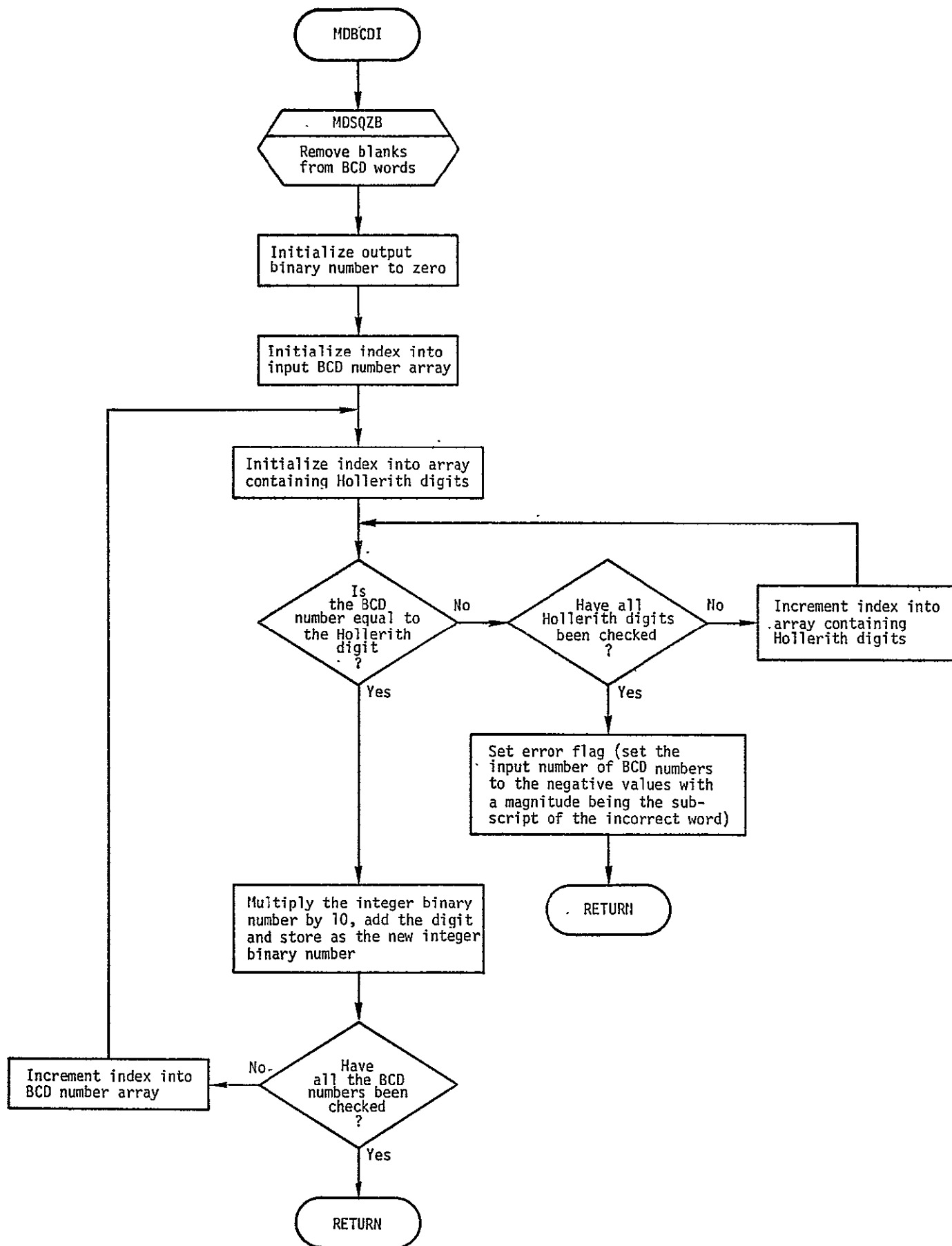
RESTRICTIONS
1. INPUT BCD ARRAY MUST CONTAIN ONLY DIGITS OR BLANKS
2. OUTPUT BCD ARRAY MUST BE LARGE ENOUGH TO CONTAIN BCD
NUMBERS

DIAGNOSTICS
NONE

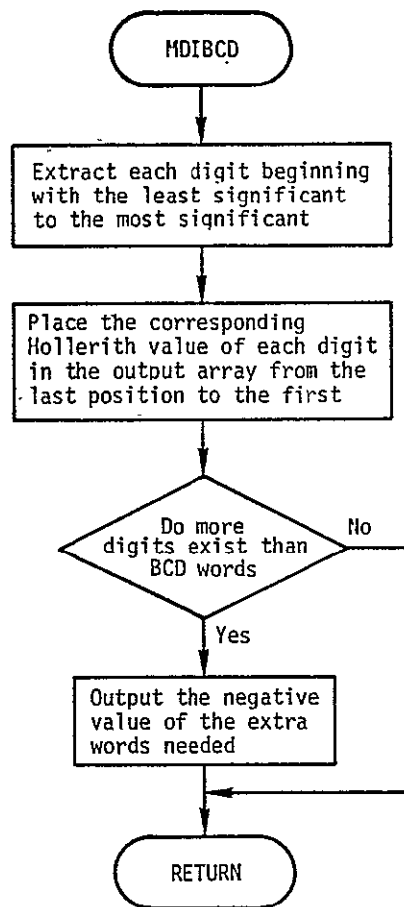
EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE



MDBC DI Flow Diagram



MDIBCD Flow Diagram

MDBCI2 - User Communication

MDBCI2 converts a BCD number to a binary integer

Method

Input: The inputs to MDBCI2 are BCD numbers one character per word to be converted and the number of characters to convert

Processing: BCD characters are input one character per word. MDBCI2 checks each word for a digit (0 - 9). If the word does not contain a digit, an error flag is set to the negative of the index to the word in error and the routine returns. The input number of words, i.e., BCD numbers in the input array, are converted to a binary output integer and the routine returns. MDBCI2 performs the same function as MDBCDI except MDBCDI allows intervening blanks in the input BCD character string.

Output: The outputs from MDBCI2 are the binary integer and an error flag if a non-digit was input in the BCD array.

USAGE

ENTRY MDBC12
CALL MDBC12 (NCOL,INT,N)

ARGMT	I/O	TYPE	DIM	DEFINITION
NCOL	I	I	N	ARRAY CONTAINING BCD NUMBERS ONE CHARACTER PER WORD
INT	O	I	1	BINARY INTEGER
N	I/O	I	1	NUMBER OF BCD WORDS INPUT IF A NON-DIGIT WAS INPUT IN ARRAY NCOL, ON OUTPUT N WILL BE A NEGATIVE VALUE WITH THE MAGNITUDE BEING THE SUBSCRIPT OF THE INCORRECT WORD

EXTERNAL REFERENCES
NONE

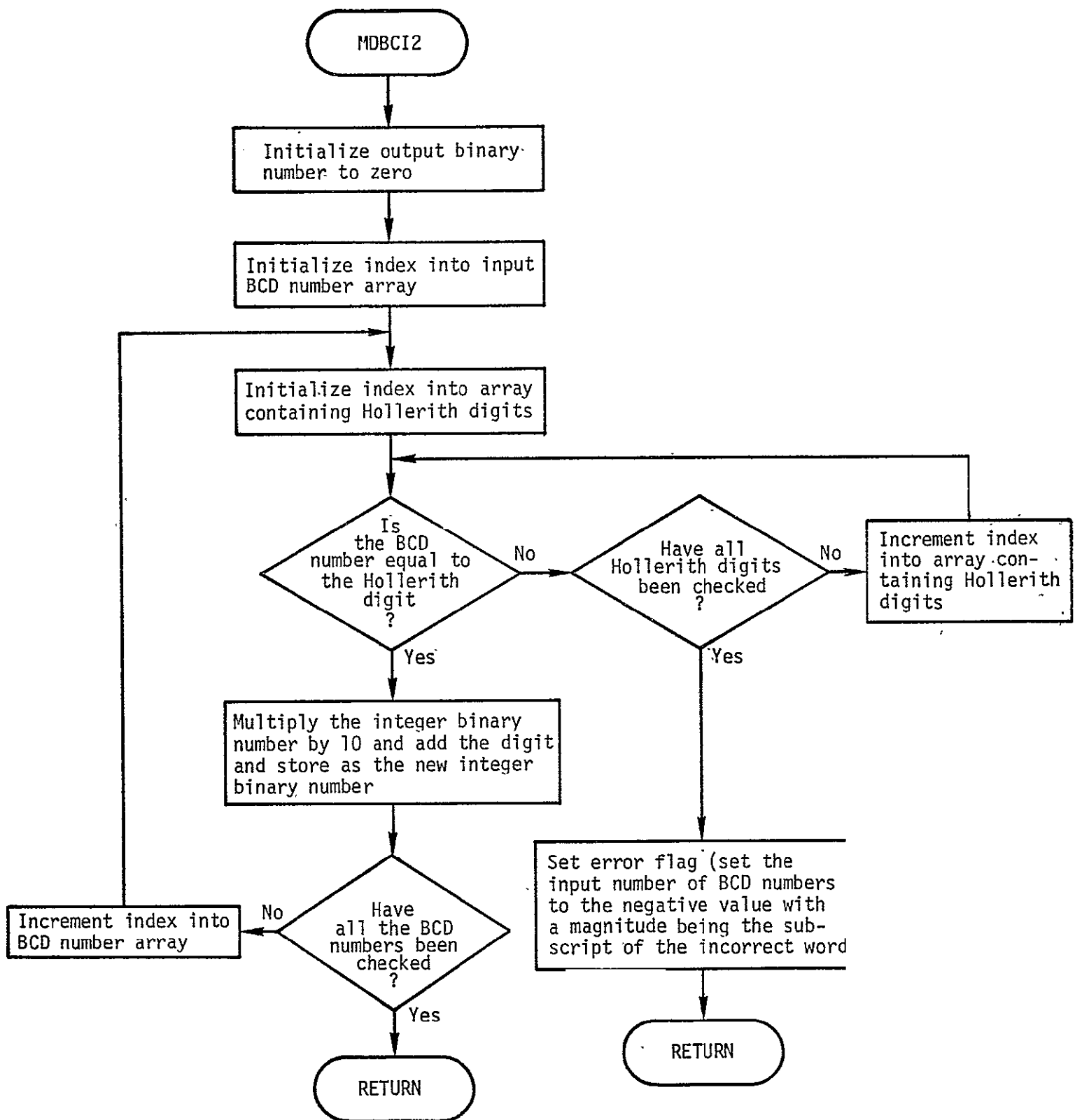
RESTRICTIONS
1. INPUT BCD ARRAY MUST CONTAIN ONLY DIGITS (0-9).

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE



MDBC12 Flow Diagram

MDCDAT - User Communications

The purpose of MDCDAT is to interpret free field card input for input data to the right of an equal sign.

Method

Input: Input to MDCDAT is through the calling sequence. It consists of a buffer containing the user's response, the pointer to the character from where processing is to begin, the number of characters in the buffer, and the column number where the prompt ended.

Processing: MDCDAT processes data on the right side of an equal sign. Integer, real, double precision, Hollerith or octal data may be input. Values that consist only of digits and prefix algebraic signs will be interpreted as integer data. MDBC DI will be called to convert the BCD number to binary integer before the value is stored in the output buffer. Numerical values will be interpreted as real (single precision) data if they contain a decimal point (.), an imbedded algebraic sign and/or the letter E. The presence of an imbedded letter D in a numerical value denotes a double precision value. Both single and double precision values will be processed with regards for underflow and overflow. Octal values are composed of the digits 0 - 7 (maximum of 12 characters) prefixed by the letter O. MDSQZB will be called to remove blank character words before the octal characters are packed into one computer word. Positive (+) and negative (-) signs will be processed for numeric values and exponents. For each numeric type, the field designator (see Appendix D), entry length and data will be stored in the output buffer.

Most other characters will be interpreted as Hollerith data. All data enclosed by apostrophes, or by a leading apostrophe and column 73, will be processed as Hollerith data. This data will be packed for output via calls to MDPCK, and thus will be left justified containing six characters per word with blank fill. The number of characters stored will be determined by the number of columns between the delimiting apostrophes or, in the absence of apostrophes, by the number of columns in the string beginning with the first non-blank common and ending with the last non-blank column (imbedded blanks are retained).

Special characters \$ and % are recognized by MDCDAT and will have the appropriate field designator set in the output buffer. A comma (,) is recognized as a field separator while an asterisk (*) and a backslash (\) are recognized as the end of statement. A left parenthesis (()) is recognized as the start of a subscript. Subscripts may contain alpha or numeric characters. The "\$LAST" feature is processed by MDCDAT; however, it is a design feature only and will not be used operationally.

MDCDAT makes numerous error checks, and outputs error messages when an error is encountered. An up arrow (^) will point to the character in error. Listed below are examples illustrating typical data forms:

- INT = 123 * (Integer)
- RVAL = 1.23E1, 45E-01 * (Real)
- DVAL = 1.23D1, 45D-01 * (Double Precision)
- BIT = 01234567 * (Octal)
- X(1) = 1.2, (2) 3.4 * (Subscript)
- ABC = 3R 123 * (Repeat)
- HD = 'HOLLERITH DATA' (BCD String)

Output: The output from MCDATA consists of error messages, an output buffer containing field designators and related data, a pointer to the end of this buffer, and a flag indicating the status of its processing.

USAGE

ENTRY MDCDAT

CALL MDCDAT (ICOM,I,NEND,ILENP,KRBF,J,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
ICOM	I	I	VARB	INPUT BUFFER CONTAINING NUMERIC VALUES TO BE CONVERTED AND PACKED
I	I	I	I	STARTING CHARACTER LOCATION OF DATA
NEND	I	I	I	NUMBER OF CHARACTERS IN ICOM
ILENP	I	I	I	NUMBER OF CHARACTERS IN PROMPT
KRBF	I	I	VARB	OUTPUT BUFFER
J	I	I	I	POINTS TO THE END OF KRBF
STAT	O	I	I	STATUS FLAG FOR MDCDAT PROCESSING
				0=NORMAL RETURN
				NEG=ERROR

EXTERNAL REFERENCES

MDBCDI

MOPCK

MDSQZ8

DIAGNOSTICS

DIGITS IN AN OCTAL VALUE, MAXIMUM OF 12 ALLOWED

NUMBER OF DIGITS FOR THE OCTAL VALUE EXCEEDED 12

ILLEGAL USE OF CHARACTER . IN OR AFTER COLUMN...

THE CHARACTER NAMED WAS USED ILLEGALLY IN OR AFTER THE COLUMN NUMBER DESIGNATED

EXTERNAL STORAGE

NONE

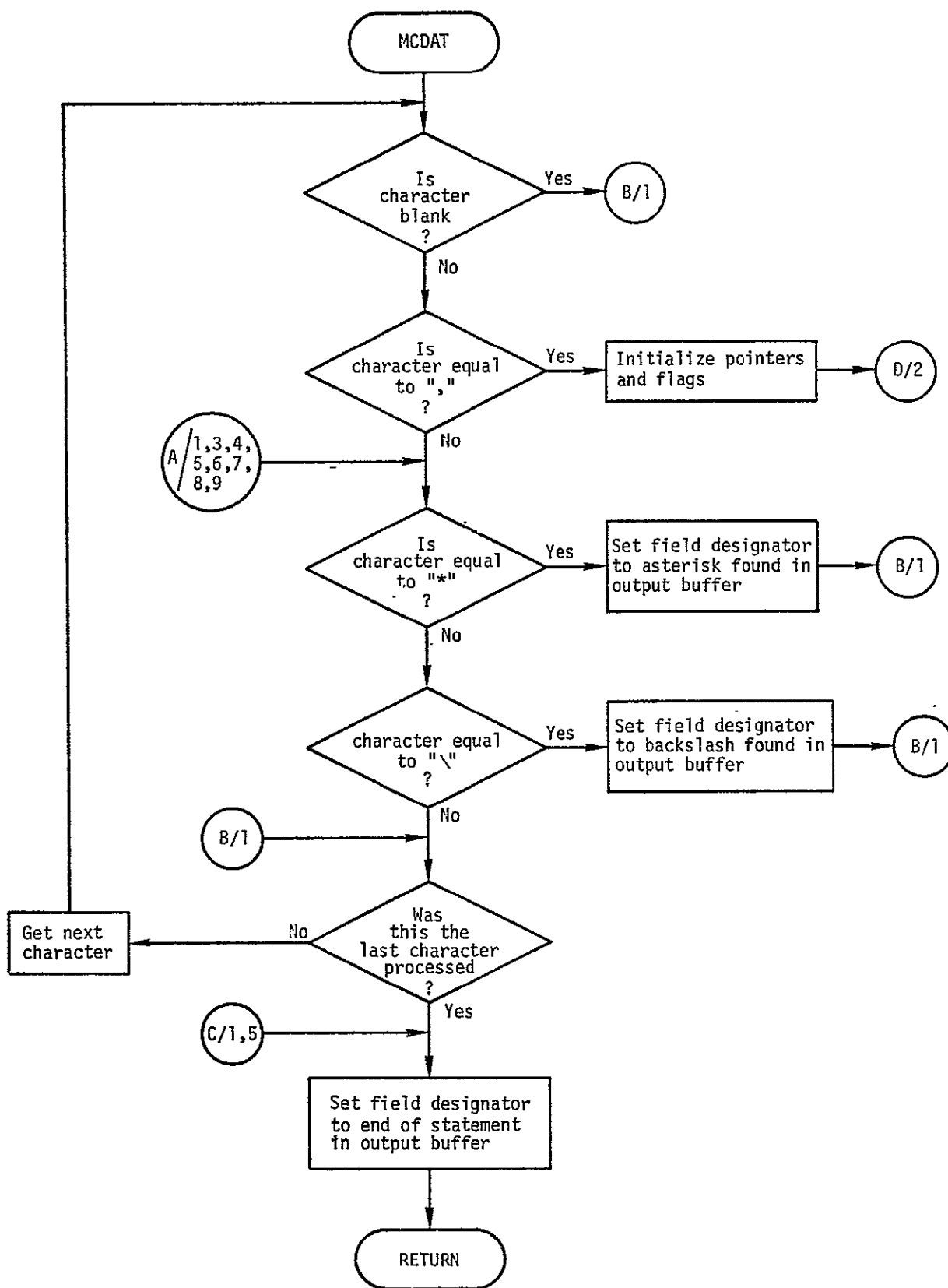
BLANK COMMON

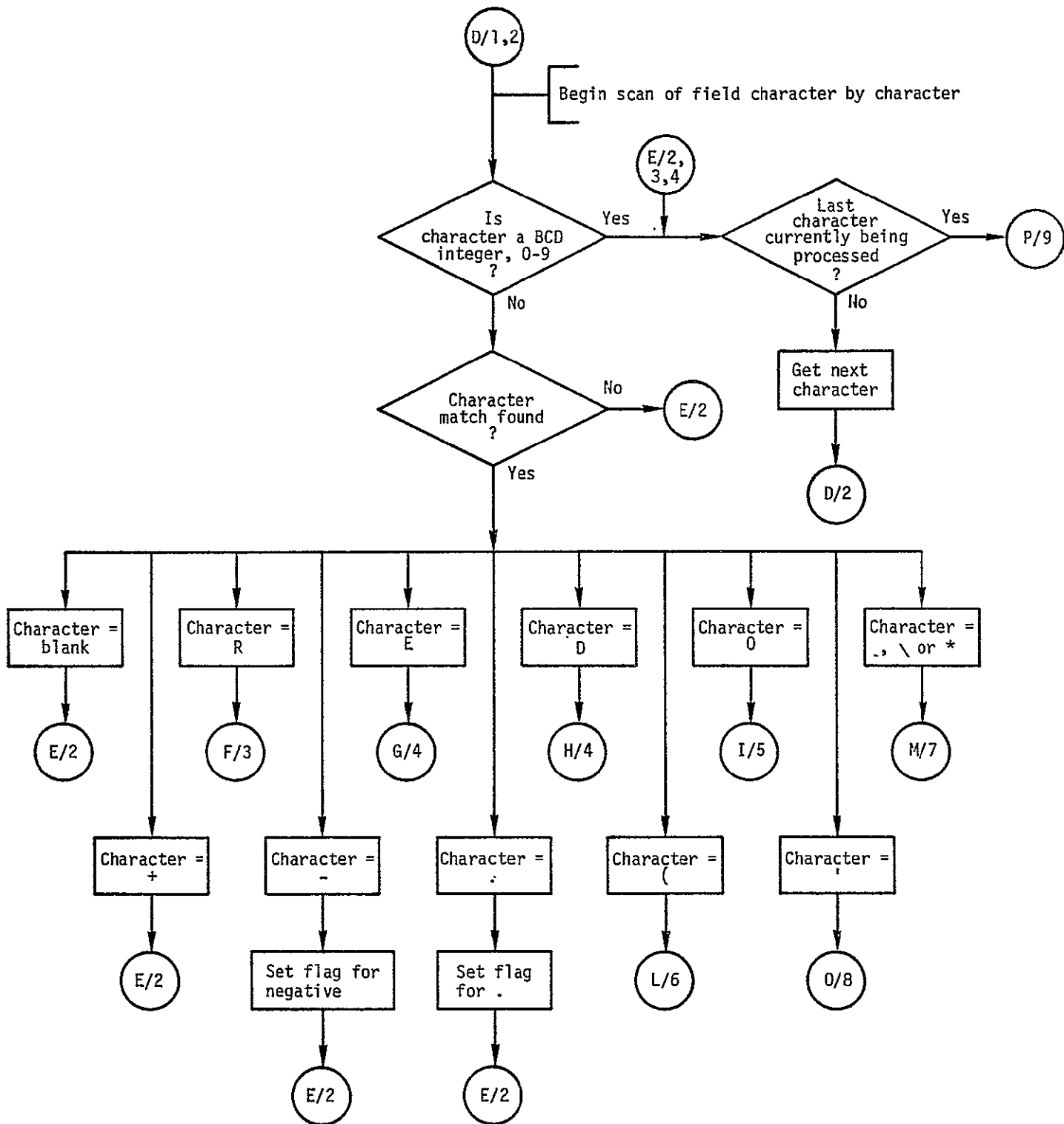
NONE

LOCAL COMMON

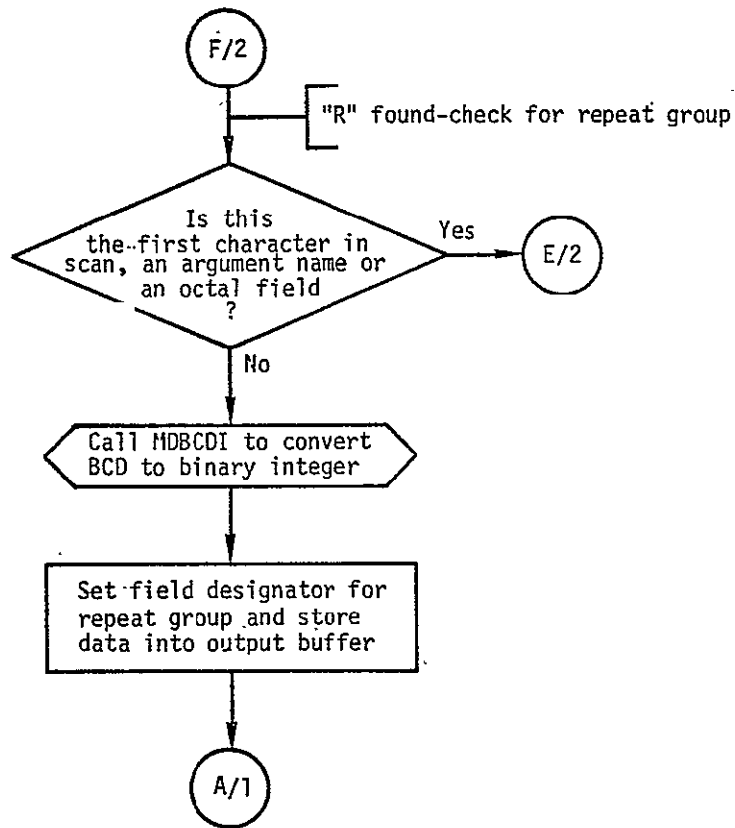
NONE

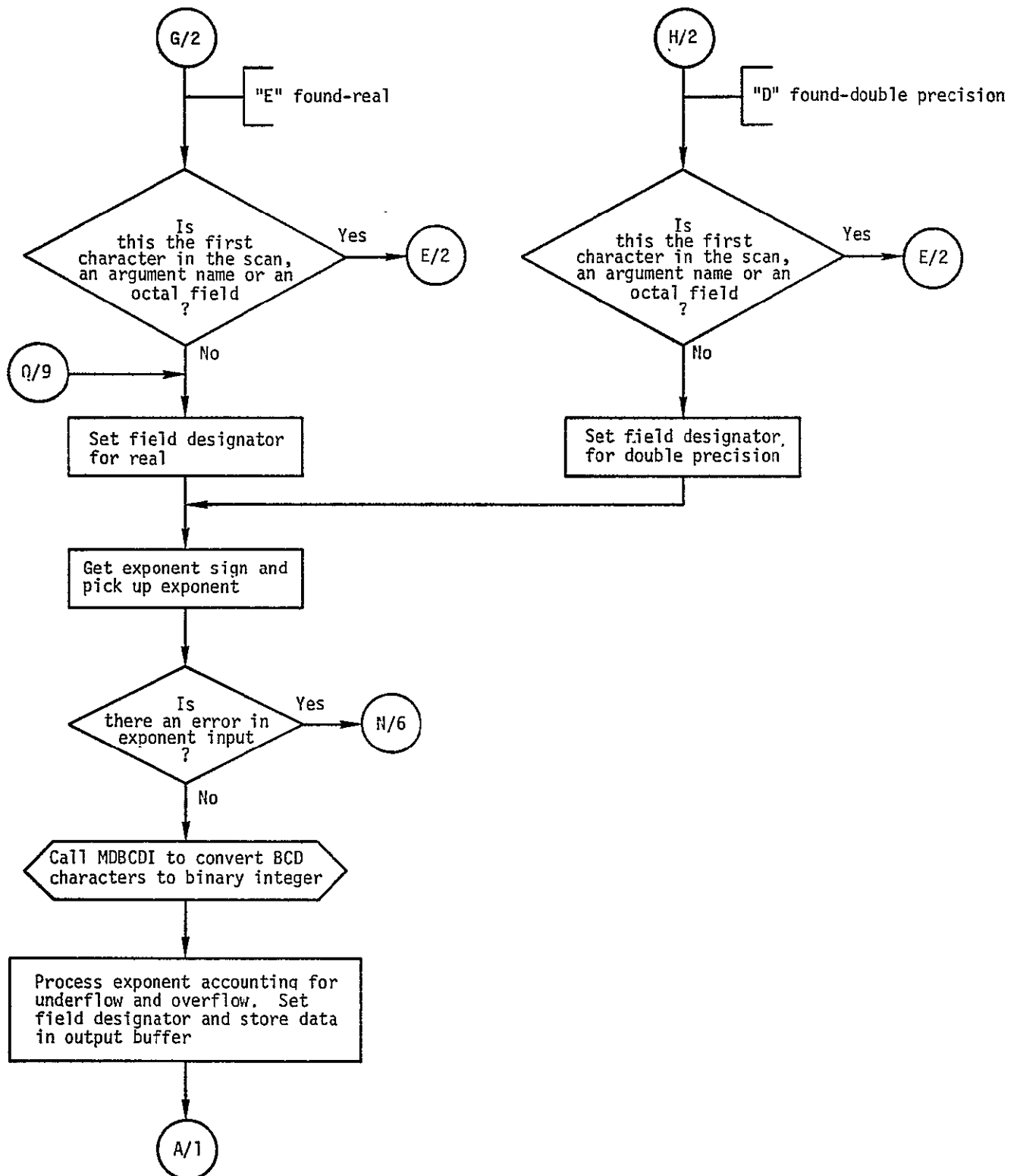
REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

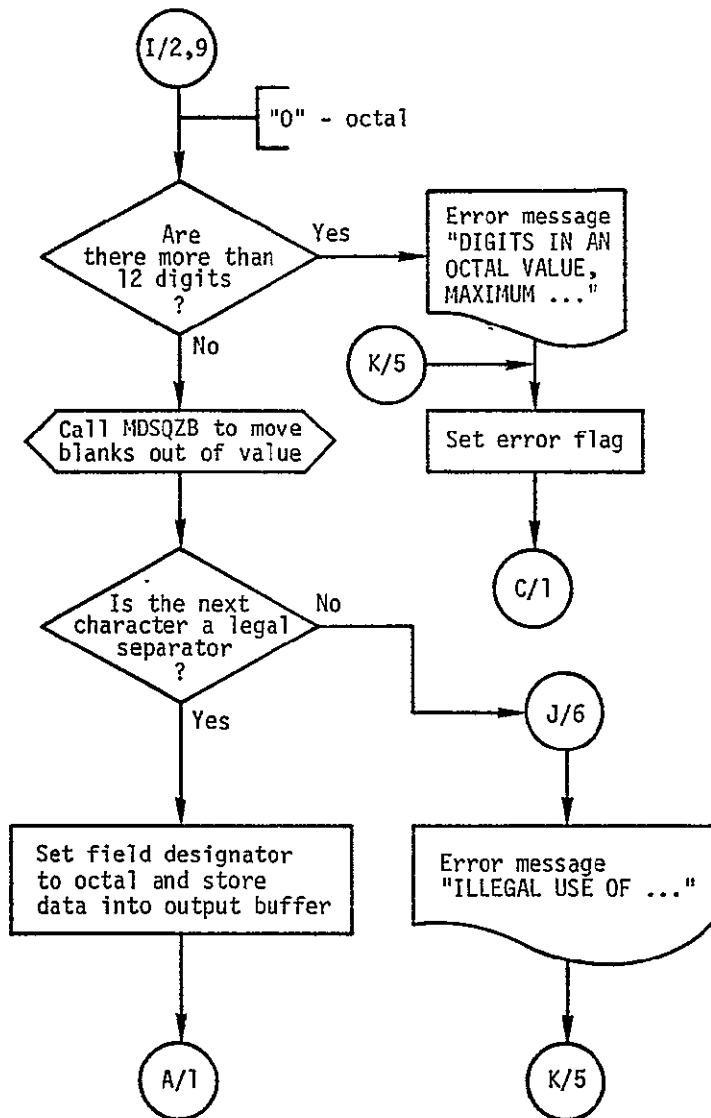


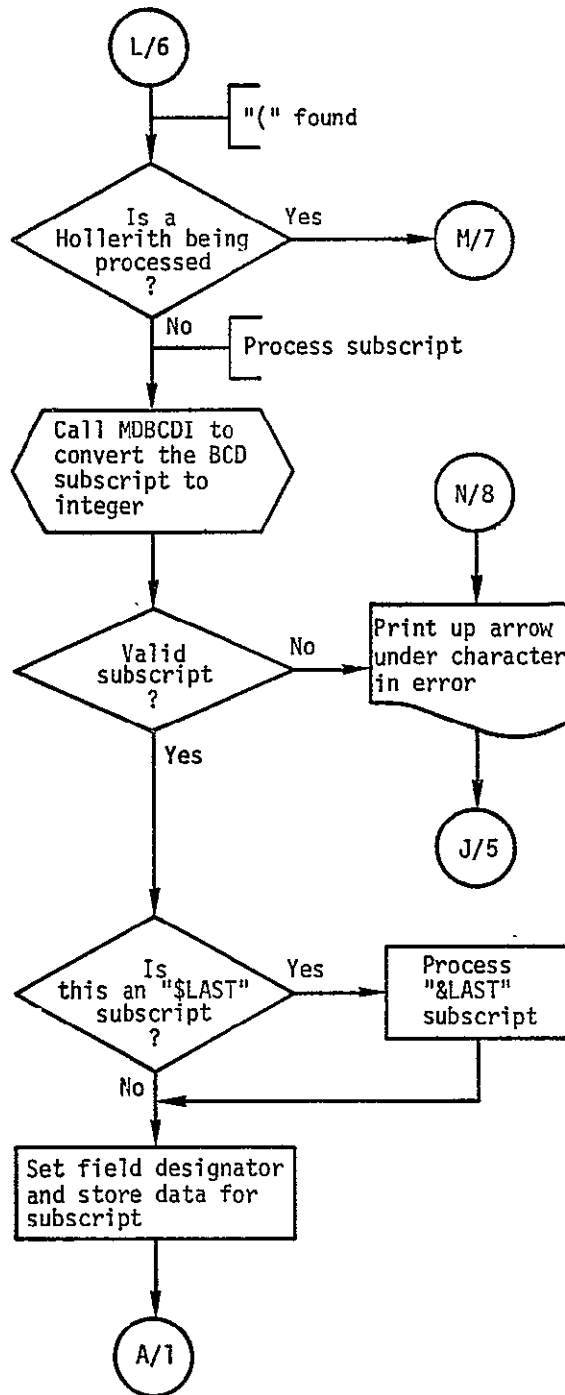


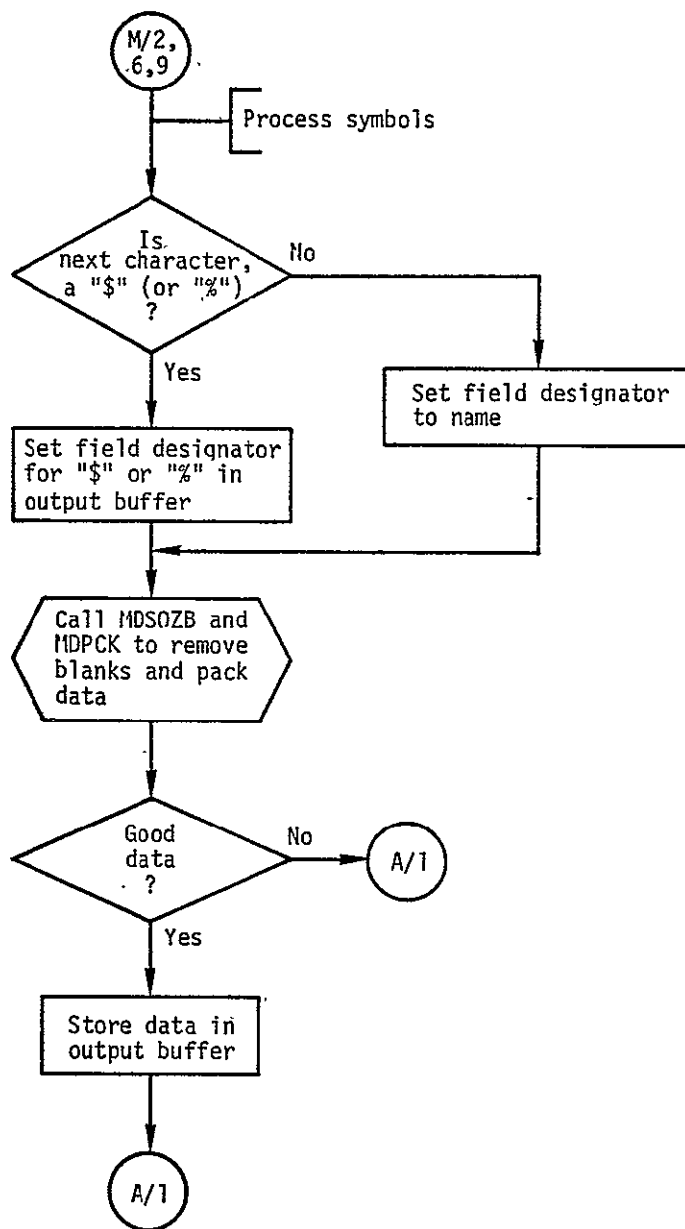
REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

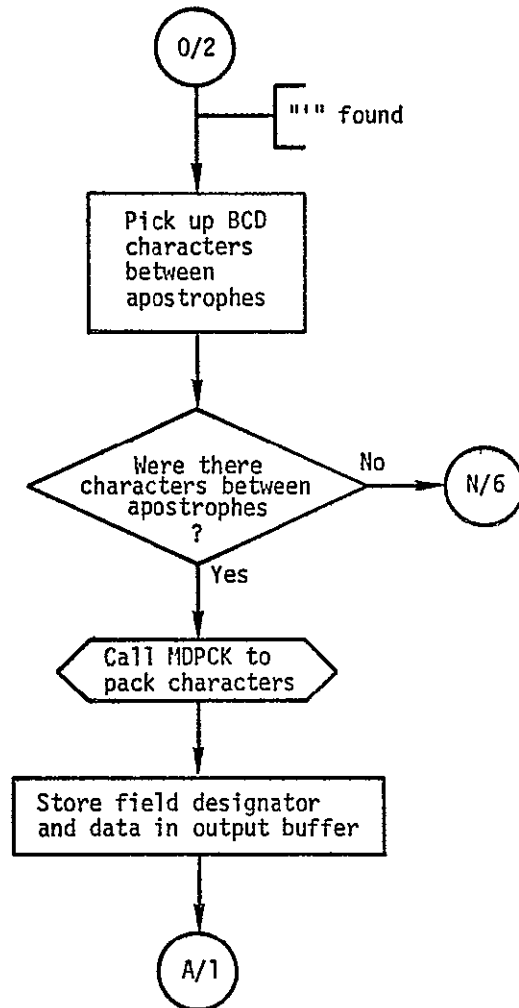


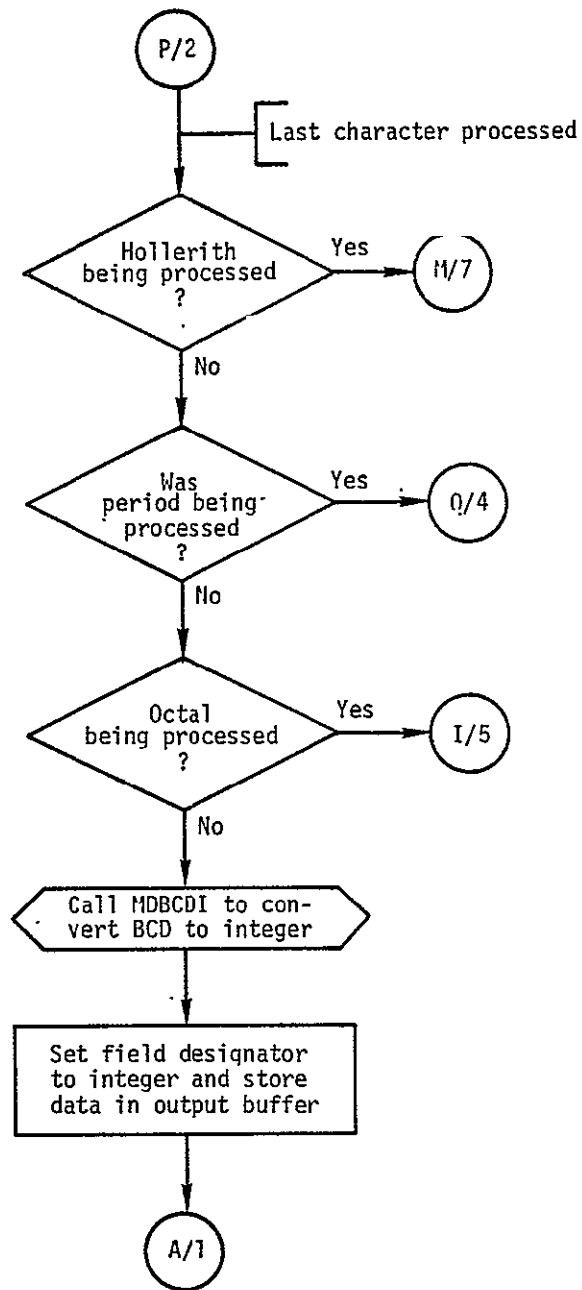












MDCONV - User Communications

MDCONV converts a BCD character string to an ASCII character string.

Method

Input: The inputs to MDCONV are the BCD buffer, the number of characters in that buffer to be converted, and the position in the output buffer to begin to place the converted characters.

Processing: MDCONV determines the starting bit of the output buffer to store the ASCII character and initializes the starting bit of the BCD buffer to zero. For each character to be converted, MDCONV determines the word number of the character in the BCD buffer and the word number to place the converted character in the ASCII buffer. An index into an array containing ASCII code is computed by extracting six bits from the BCD buffer starting at the specified bit. The nine bit ASCII character is placed in the ASCII buffer. The bit location of ASCII is incremented by 9 and the bit location of the BCD buffer is incremented by 6. After all characters have been converted, MDCONV returns.

Output: The output from MDCONV is a buffer containing the ASCII characters. This buffer is not affected other than the ASCII character string has been inserted.

USAGE

ENTRY MDCONV.
CALL MDCONV (N,BCDBUF,ICHAR,ASCBUF)

ARGMT	I/O	TYPE	DIM	DEFINITION
N	I	I	I	NUMBER OF CHARACTERS TO BE CONVERTED
BCDBUF	I	I	VARB	BUFFER CONTAINING THE BCD CHARACTERS
ICHAR	I	I	I	CHARACTER POSITION IN ASCII BUFFER TO BEGIN OUTPUT STRING
ASCBUF	O	I	VARB	ASCII BUFFER CONTAINING THE OUTPUT CHARACTER STRING (NOTE: ASCBUF IS NOT AFFECTED OTHER THAN THE INSERTED STRING)

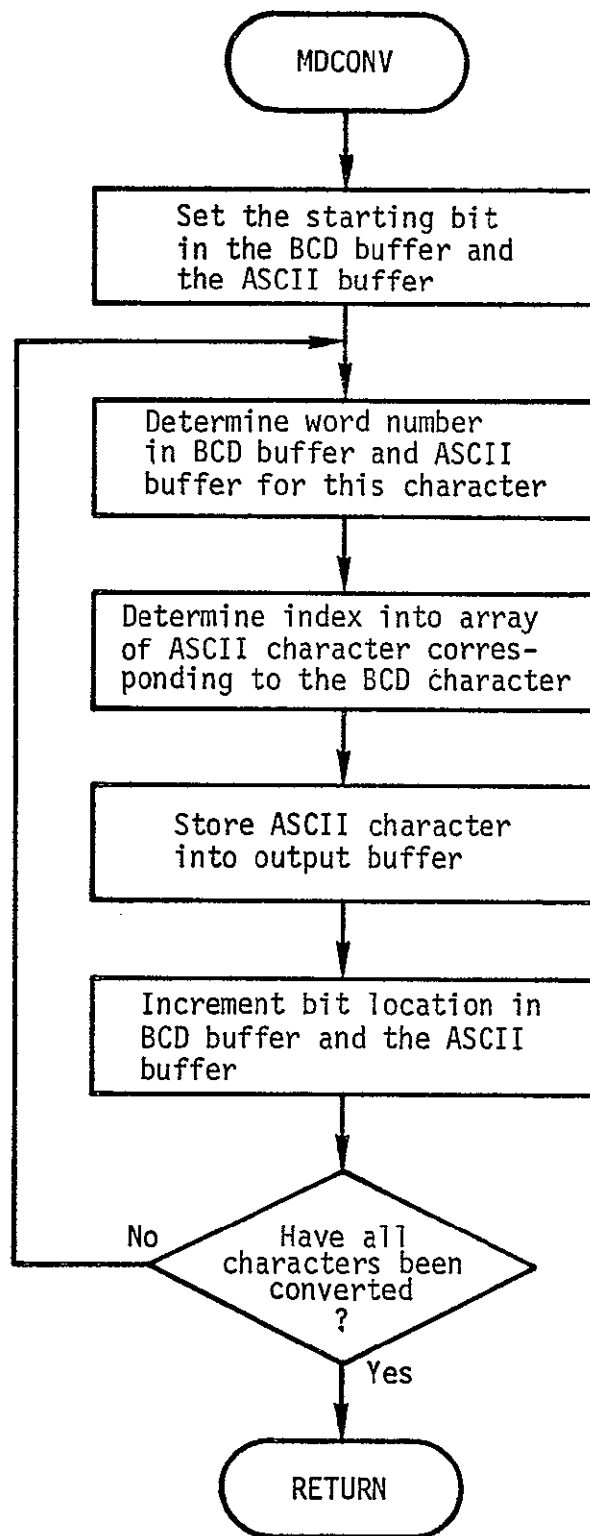
EXTERNAL REFERENCES
NONE

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE



MDCONV Flow Diagram

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDPCK - User Communication

MDPCK packs characters from a single character per word array into a six character per word array.

Method

Input: The inputs to MDPCK are the number of characters to pack and the array containing the characters to be packed.

Processing: MDPCK determines the number of words needed to pack the input characters. MDPCK then loops storing six characters at a time into the output array. During the store, the first bit of the first character in the six character set is removed before the store and then returned after the store. This is done to prevent overflow during packing. If the last word is not filled with input characters, the remaining characters in the output word will be blanks.

Output: The outputs from MDPCK are the array containing the packed characters and the number of words in that array.

SAGE.

ENTRY MDPCK
CALL MDPCK (NCARDS,LWORD,N)

ARGMT	I/O	TYPE	DIM	DEFINITION
NCARDS	I	I	VARB	ARRAY CONTAINING ONE CHARACTER PER WORD
LWORD	O	I	VARB	ARRAY CONTAINING PACKED CHARACTERS (SIX CHARACTERS PER WORD)
N	I/O	I	I	ON INPUT N IS THE NUMBER OF SINGLE CHARACTERS (DIMENSION OF NCARDS) ON OUTPUT N IS THE NUMBER OF WORDS OF PACKED CHARACTERS (SIX CHARACTERS PER WORD)

EXTERNAL REFERENCES
NONE

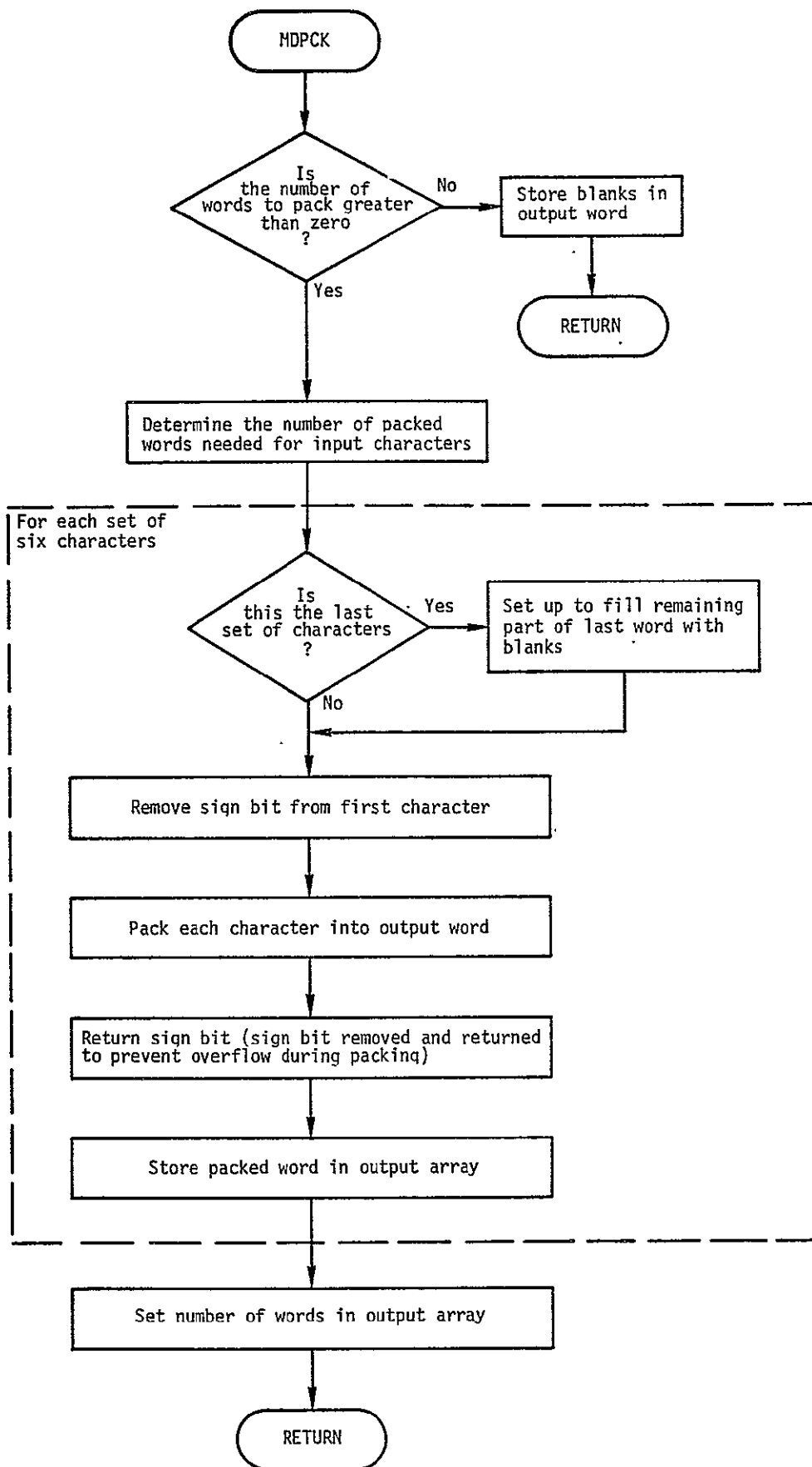
DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDPCK Flow Diagram

MDPRMH - User Communications

The purpose of MDPRMH is to prompt the user with the variable name and associated Hollerith values, and to return the response.

Method

Input: The input to MDPRMH consists of a Hollerith variable name and associated Hollerith values input through the calling sequence.

Processing: The variable name and values are stored in the buffer passed to MDPRMT (which prompts the user for a response). If the number of computer words exceeds the print line, then MDPRMH will print all lines except the last line which will be printed by MDPRMT as the prompt for the user's response.

Output: The output is a buffer containing the user's response and the status of this output.

USAGE

ENTRY MDPRMH

CALL MDPRMH (NAME,ARRAY,LEN,BUFF,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
NAME	I	H	1	ITEM CONTAINS THE VARIABLE NAME
ARRAY	I	H	VARB	ARRAY CONTAINS HOLLERITH VALUES ASSOCIATED WITH NAME
LEN	I	I	1	NUMBER OF WORDS CONTAINING ARRAY DATA
BUFF	O	I	VARB	BUFFER CONTAINING USER RESPONSE-UNITS-INTERNAL BCD
STAT	O	I	1	STATUS FLAG FOR MDPRMH PROCESSING

EXTERNAL REFERENCES
MDPRMT

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
VARB I/O

NONE

LOCAL COMMON	VARB	I/O	TYPE	DIM	LOC. RELADD	DEFINITION
NONE						

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDPRMI - User Communications

The purpose of MDPRMI is to prompt the user with the variable name and associated integer values, and to return the response.

Method

Input: The input to MDPRMI consists of a Hollerith variable name and associated integer values input through the calling sequence.

Processing: MDPRMI prompts the user for a response, via MDPRMT, to the requested input integer values. If the number of values exceeds one print line, MDPRMI prints all lines except the last one, which is sent to MDPRMT as the prompt.

Output: The output is a buffer containing the user's response and the status of this output.

USAGE

ENTRY MDPRMI

CALL MDPRMI (NAME,ARRAY,LEN,BUFF,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
NAME	I	H	I	ITEM CONTAINS THE VARIABLE NAME
ARRAY	I	I	VARB	ARRAY CONTAINS INTEGER VALUES ASSOCIATED WITH NAME
LEN	I	I	I	NUMBER OF WORDS CONTAINING ARRAY DATA
BUFF	O	I	VARB	BUFFER CONTAINING USER RESPONSE UNITS=INTERNAL BCD
STAT	O	I	I	STATUS FLAG FOR MDPRMI PROCESSING

EXTERNAL REFERENCES

MDPRMT

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

VARB I/O

NONE

LOCAL COMMON

VARB	I/O	TYPE	DIM	LOC	RELADD	DEFINITION
------	-----	------	-----	-----	--------	------------

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDPRMR -- User Communications

The purpose of MDPRMR is to prompt the user with the variable name and associated real values, and to return the response.

Method

Input: The input to MDPRMR consists of a Hollerith variable name and associated real values input through the calling sequence.

Processing: If there is less than one print line of real values associated with the variable name input to MDPRMR, MDPRMT is called with these values to prompt the user for a response. If one line is exceeded, MDPRMR prints the Hollerith name and all the values other than the last one which is passed to MDPRMT as the prompt for the user's response.

Output: The output is a buffer containing the user's response and the status of this output.

USAGE

ENTRY MDPRMR
CALL MDPRMR (NAME,ARRAY,LEN,BUFF,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
NAME	I	H	1	ITEM CONTAINS THE VARIABLE NAME
ARRAY	I	R	VARB	ARRAY CONTAINS REAL VALUES ASSOCIATED WITH NAME
LEN	I	I	1	NUMBER OF WORDS CONTAINING ARRAY DATA
BUFF	O	I	VARB	BUFFER CONTAINING USER RESPONSE * UNITS-INTERNAL BCD
STAT	O	I	1	STATUS FLAG FOR MDPRMR PROCESSING

EXTERNAL REFERENCES
MDPRMT

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
VARB I/O

NONE

LOCAL COMMON	VARB	I/O	TYPE	DIM	LOC	REL-ADD	DEFINITION
NONE							

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDPRMT - User Communications

The purpose of MDPRMT is to provide the submonitor prompting capability.

Method

Input: The input to MDPRMT consists of a field data character string, number of characters in the string and a flag specifying the scan type.

Processing: MDPRMT converts the internal BCD characters to ASC II via MDCONV, adds an end mark and a null character, and prints this data as the prompt for the user. The response is read and interpreted to internal BCD via MDSCAN. If an up arrow "↑" was input, the routine will prompt with "↑" until some other response is input before returning to the caller.

Output: The output is a buffer containing the format of each field encountered in the one line text input by the caller.

USAGE

ENTRY MDPRMT

CALL MDPRMT (PRMT,L,EQUFLG,INPUT,STATUS)

ARGMT	I/O	TYPE	DIM	DEFINITION
PRMT	I	I	VARB	FIELD DATA CHARACTER STRING
L	I	I	I	NUMBER OF CHARACTERS IN PRMT
EQUFLG	I	I	I	SCAN TYPE FLAG
INPUT	O	I	VARB	DECODED INPUT LINE (DECODED BY MDSCAN)
STATUS	O	I	I	STATUS FLAG FOR MDPRMT PROCESSING

EXTERNAL REFERENCES

MDCONV

MDSCAN

DIAGNOSTICS

ABOVE INPUT LINE HAS ILLEGAL CHARACTERS--PLEASE CORRECT AND REINPUT

AN ILLEGAL CHARACTER WAS FOUND WHEN THE CHARACTER STRING WAS SCANNED

EXTERNAL STORAGE

NONE

BLANK COMMON

NONE

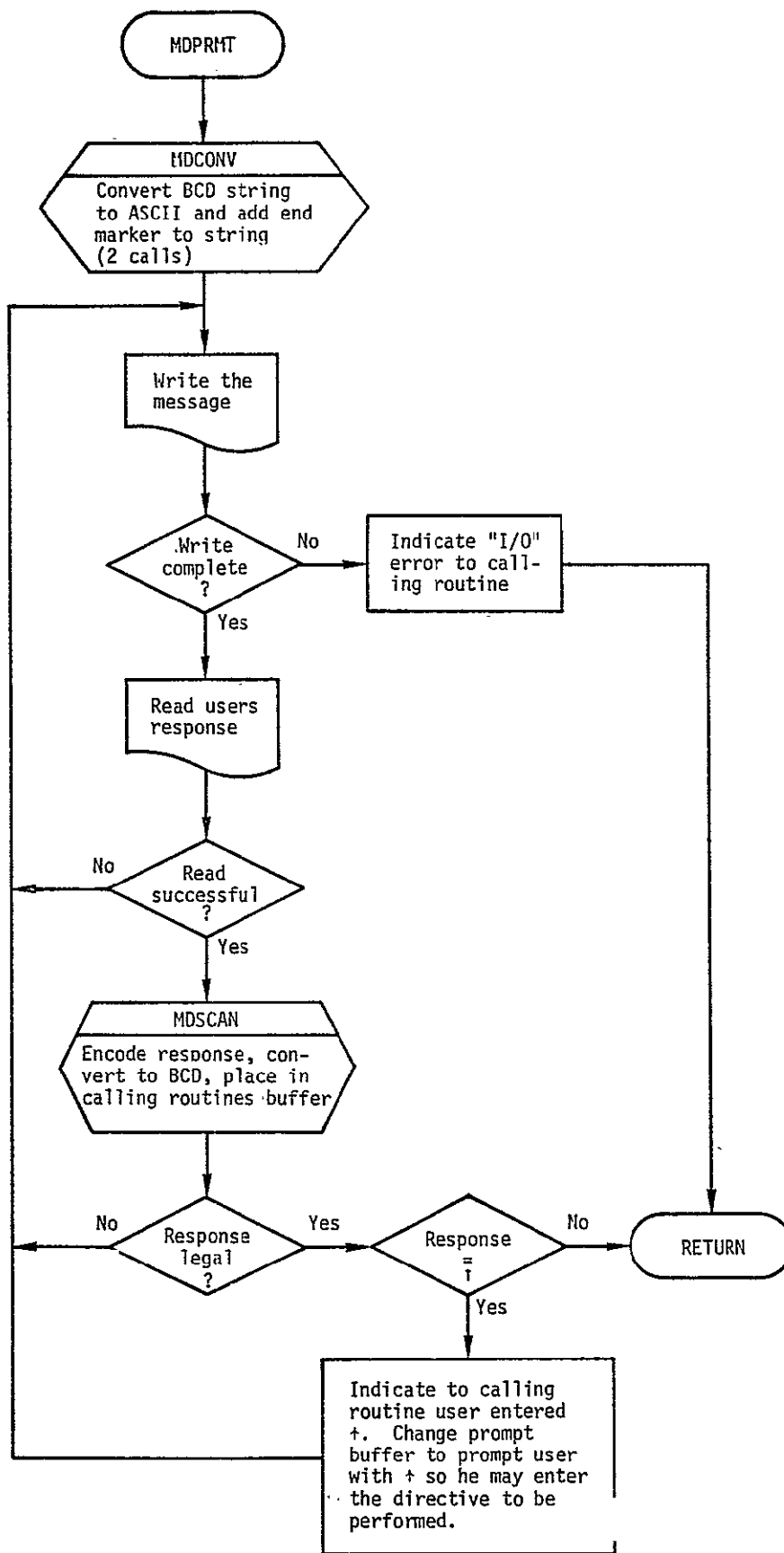
COMMON /MDCODE/

APOST	0
ASTRSK	0
AT	0
BCKSLH	0
COLON	0
COMMA	0
DBLE	0
DOLLAR	0
EOS	0
EQUALS	0
HOLL	0
INTEG	0
LBSIGN	0
LPAR	0
MINUS	0
NAME	0
OCTAL	0
PERCNT	0
PLUS	0
QUESMK	0
REAL	0
RPAR	0
REPEAT	0
SLASH	0
SUBS	0
UPARRW	0

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

LOCAL COMMON

NONE



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDPRMT Flow Diagram

MDSCAN - User Communications

The purpose of MDSCAN is to scan each field of the text line and output an encoded buffer. This buffer contains information relating each field encountered in the text via a numerical code followed by the data values.

Method

Input: The input to MDSCAN consists of a buffer (in BCD), number of characters in the buffer, a flag specifying the scan type, and a flag containing the number of characters contained in the prompt. This data is input through the calling sequence.

Processing: MDSCAN translates the input text line into fields for the output buffer. Values on the right side of an "=" sign are interpreted and packed into the output buffer by MDCDAT. Numeric values not following an equal sign are converted via MDBC12 to binary integer. MDPCK is called to pack binary integers or alpha characters (whichever processing is occurring) into a character string to be stored by MDSCAN in the output buffer.

For subscript values, numeric subscripts are first converted to binary integer via MDBC12 before MDPCK is called to pack these digits. Since alphabetic characters do not require this conversion, MDPCK is called immediately. The packed characters are then stored by MDSCAN into the output buffer. Subscripts for more than one dimensional array will also be processed. If the subscript request was for "&LAST", special processing will occur. The handling of "&LAST" is a design feature and will not be considered in detail because it will not be used operationally.

When MDSCAN builds the output buffer, the field designator, and entry length and data (if applicable) are stored in the output buffer for each field encountered. For definition of the field designators, refer to the appendix. If an error occurred during processing, the status flag is set to the character found to be in error. If no errors were encountered, the status is set to 0.

Output: The output from MDSCAN consists of a buffer containing the field designator and, if applicable, the entry length and data for each field in the input text line (see Appendix for details). Also output is the status of MDSCAN's processing. These parameters are output through the calling sequence.

USAGE

ENTRY MDSCAN

CALL MDSCAN (INPUT,N,EQUFLG,PRTLEN,BUFF,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
INPUT	I	I	VARB	UPON ENTRY INPUT CONTAINS A ONE LINE TEXT
N	I	I	1	UPON ENTRY N CONTAINS THE NUMBER OF CHARACTERS IN INPUT
EQUFLG	I	I	1	UPON ENTRY EQUFLG IF \ 0 MEANS AN EQUAL SIGN HAS BEEN PROMPTED AND A LITERAL STRING MAY FOLLOW. OTHERWISE INPUT IS A LITERAL LIST
PRTLEN	I	I	1	UPON ENTRY PRTLEN CONTAINS THE NUMBER OF CHARACTERS IN THE PROMPT
BUFF	O	I	VARB	ENCODED BUFFER CONTAINING THE FIELD DESCRIPTOR AND DATA FOR EACH FIELD IN THE TEXT LINE
STAT	O	I	1	STATUS OF THE OUTPUT. 0=NORMAL RETURN -LS 0=GIVES THE COLUMN NUMBER OF THE ERROR

EXTERNAL REFERENCES

MOBCI2
MDCDAT
MDPCK

DIAGNOSTICS

NONE

EXTERNAL STORAGE

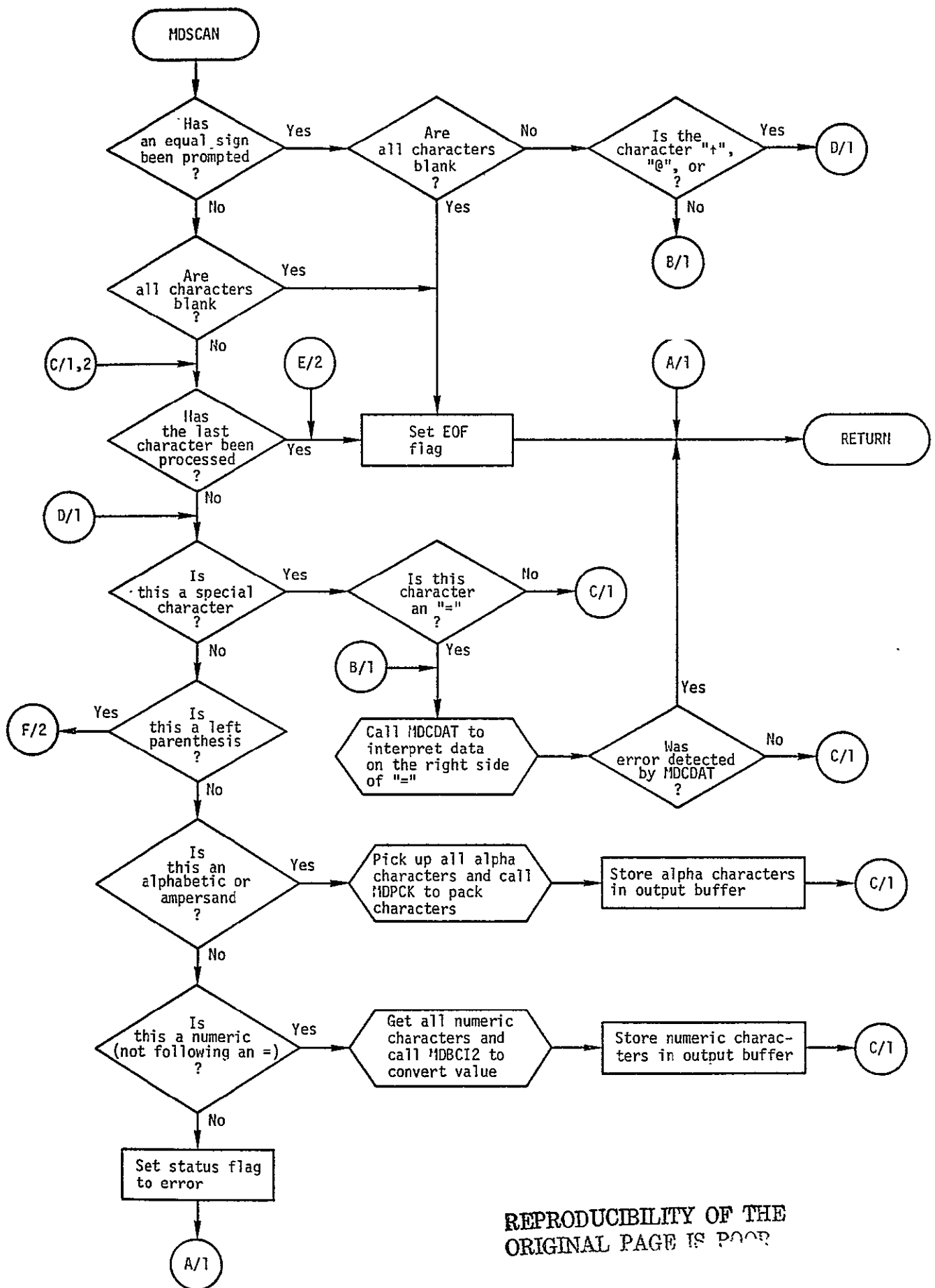
NONE

BLANK COMMON

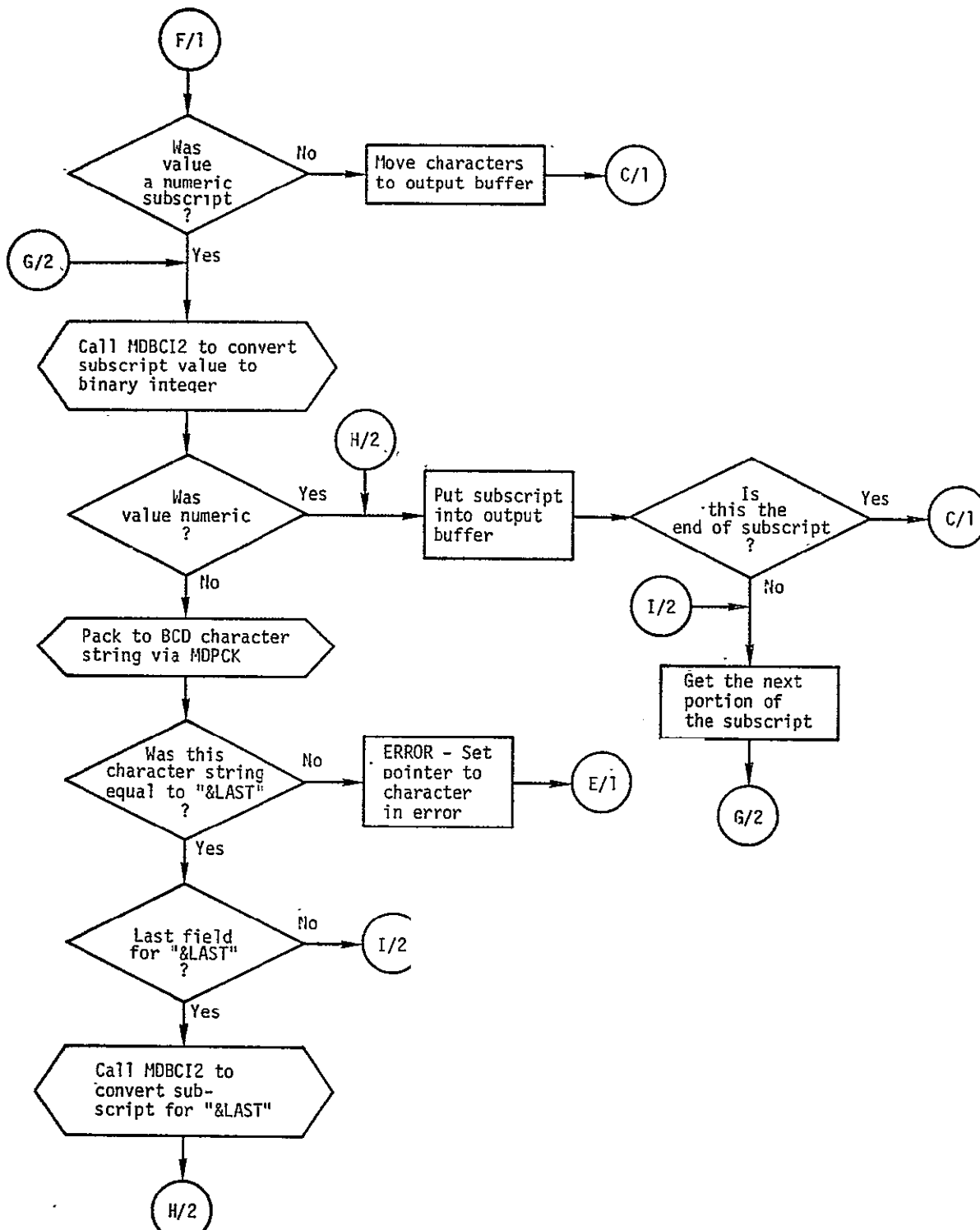
NONE

LOCAL COMMON

NONE



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDSQZB - User Communications

The purpose of MDSQZB is to remove imbedded blanks in a character string.

Method

Input: The inputs to MDSQZB are a string of characters with imbedded blanks to be removed and the number of characters in that string to search for blanks.

Processing: MDSQZB examines the specified number of characters in the input string beginning with the first character. When a non-blank character is encountered, it is stored in the output string. If the input number of characters to examine is negative, blanks within a Hollerith string will remain.

Output: The outputs from MDSQZB are a character string with no imbedded blanks and the number of non-blank characters.

ISAGE

ENTRY MDSQZB
CALL MDSQZB (NCOL,N)

ARGMT I/O TYPE DIM

DEFINITION

NCOL I/O I N

ON INPUT NCOL IS AN ARRAY OF CHAR-
ACTERS IN WHICH BLANK ARE TO BE
REMOVED.

ON OUTPUT NCOL IS AN ARRAY CONTAINING
THE PACKED CHARACTERS.

N I/O I 1

ON INPUT N IS THE NUMBER OF COLUMNS
IN INPUT ARRAY TO SEARCH FOR BLANKS.
IF NEGATIVE, BLANKS WILL NOT BE
REMOVED WITHIN HOLLERITH CHARACTERS
I.E. BETWEEN APOSTROPHES.
ON OUTPUT N IS THE NUMBER OF NON-
BLANKS CHARACTERS IN NCOL.

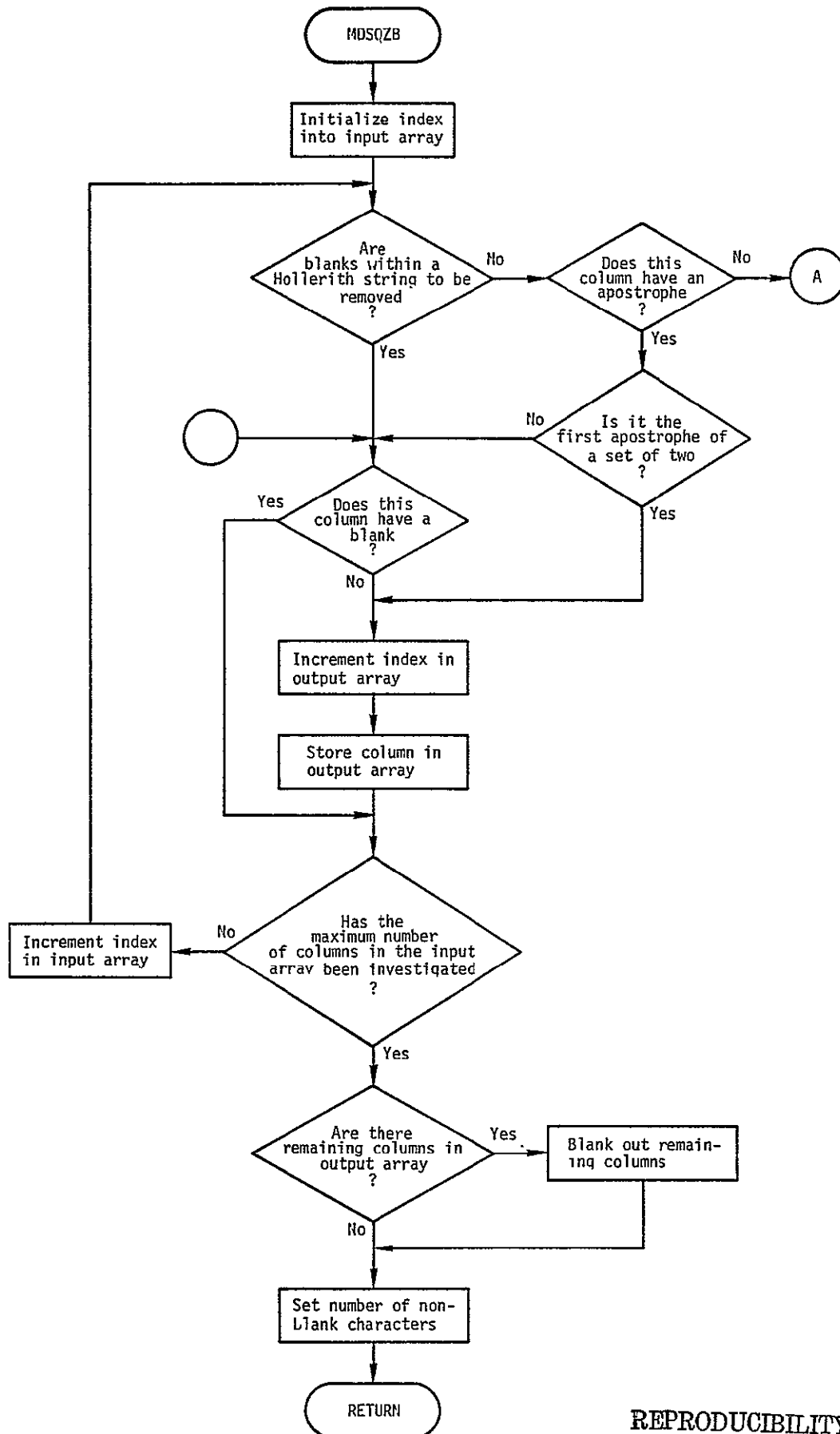
EXTERNAL REFERENCES
NONE

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDELETE - Storage Monitor

MDELETE deletes an entry from the storage monitor table (SMT)

Method

Input: The inputs to MDELETE are the label of the desired storage monitor entry to be deleted and the entry type if a search for type is to be made.

Processing: MDELETE locates the entry in the storage monitor table corresponding to the input label and flags it for deletion. The required storage is not released, however and the data base is not automatically packed by MDELETE. If MDELETE could not find the label of the SMT entry, a flag is set to indicate that the entry was not flagged for deletion.

Output: The outputs from MDELETE are a flag indicating whether the SMT entry was found and deleted or not. If the SMT was deleted, the sort, pack, and deactivate flags are output to indicate deletion of the entry.

ISAGE

ENTRY MDELET

CALL MDELET (LABEL,TYPE,NOFIND)

ARGMT	I/O	TYPE	DIM	DEFINITION
LABEL	I	I	1	LABEL OF THE DESIRED SMT ENTRY
TYPE	I	I	1	ENTRY TYPE - IF NEGATIVE NO SEARCH FOR TYPE IS MADE
NOFIND	O	I	1	ENTRY FIND FLAG =0 ENTRY WAS FOUND AND DELETED =1 ENTRY WAS NOT FOUND

EXTERNAL REFERENCES

NOFIND

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

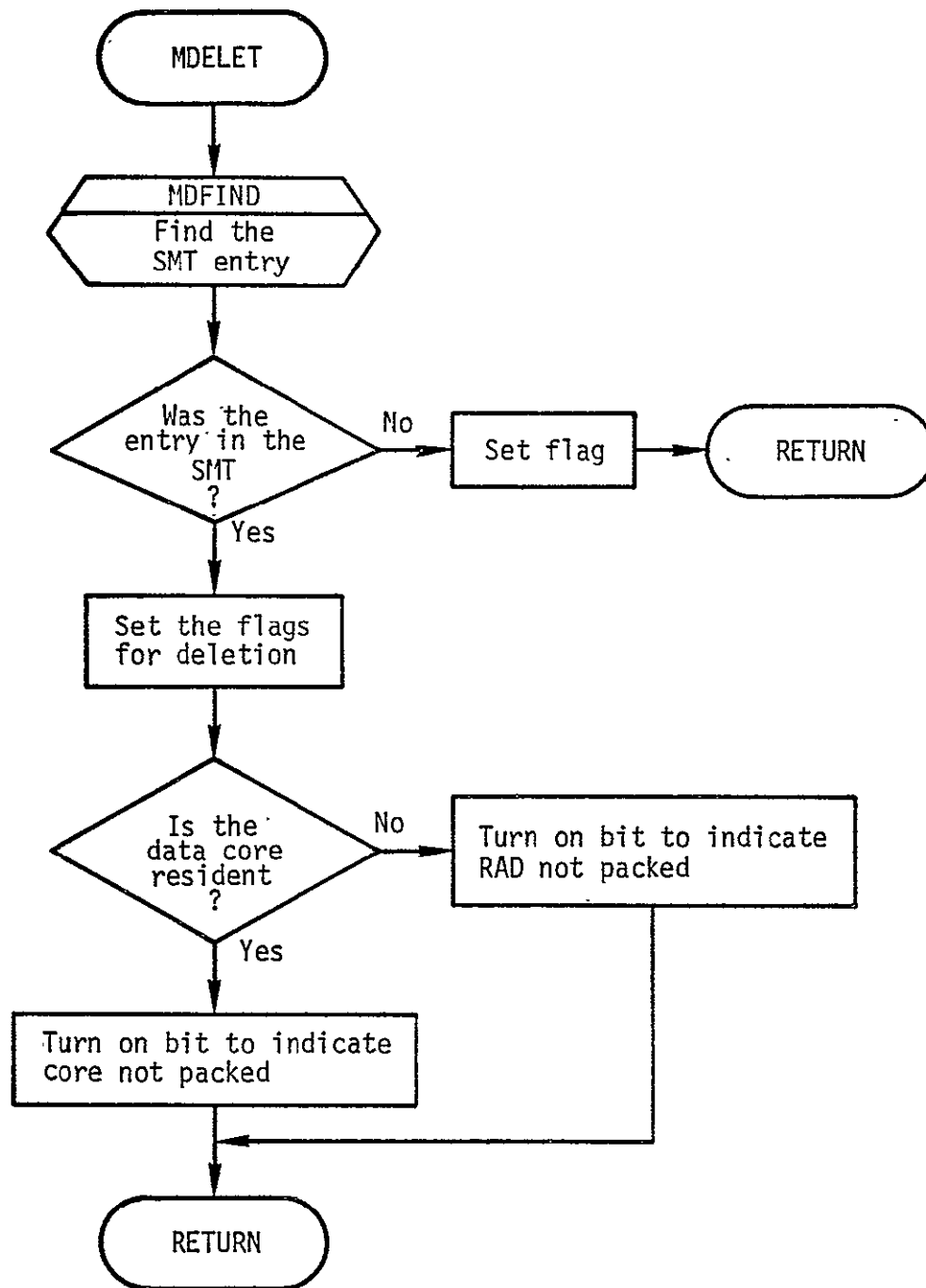
VARB I/O

SORTFG 0

PACK 0

LOCAL COMMON

NONE



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDELETE Flow Diagram

MDENTR - Storage Monitor

MDENTR is used by the MDAS monitor to allocate storage for the storage monitor table (SMT).

Method

Input: Description (type, label, length and column dimension) of the SMT entry to allocate and a flag indicating storage device (memory or RAD) are input along with current addresses and flag of the present SMT.

Processing: MDENTR builds an entry in the storage monitor table and allocates storage for the data either in core or on an external storage device. The SMT is a part of blank common area beginning at DBSTRT (a blank common variable). For a definition of an SMT entry see Section 5.3.

The core resident data is allocated from the bottom of blank common backwards toward the SMT. If storage is not available either in the SMT or the data area, the data base is packed in order to squeeze out the deleted entries. If storage is still not available an error flag is output, a message is printed and the routine returns. When an entry is placed in the SMT and storage is allocated, the sort flags are set to indicate that the SMT is not sorted.

The algorithm for MDENTR is depicted in the functional flow diagram.

Output: The address and size of the data allocation and addresses and flags associated with the new SMT are output along with an output flag.

USAGE

ENTRY MDENTR.

CALL MDENTR (TYPE,LABEL,SIZE,IDIM,ADDR,DEVICE,ERROR,

ARGMT	I/O	TYPE	DIM	DEFINITION
TYPE	I	I	I	ENTRY TYPE FLAG
LABEL	I	I	I	ENTRY LABEL
SIZE	I	I	I	LENGTH OF ENTRY
IDIM	I	I	I	COLUMN DIMENSION OF ENTRY
ADDR	O	I	I	ADDRESS OF DATA ALLOCATION FOR CORE RESIDENT DATA ADDRESS IS GIVEN RELATIVE BLANK COMMON FOR DATA RESIDING ON EXTERNAL STORAGE ADDRESS IS THE NEGATIVE ADDRESS
DEVICE	I	I	I	DEVICE INDICATOR #0 ALLOCATE CORE STORAGE #1 ALLOCATE EXTERNAL STORAGE
ERROR	O	I	I	ERROR RETURN FLAG #0 NO ERROR #1 STORAGE MONITOR TABLE IS FULL #2 CORE STORAGE EXCEEDED #3 EXTERNAL STORAGE EXCEEDED

EXTERNAL REFERENCES
MDPACK

DIAGNOSTICS
***STORAGE REQUIREMENTS EXCEEDED
THE STORAGE REQUIRED EITHER FOR CORE OR EXTERNAL DEVICES
IS GREATER THAN THAT AVAILABLE

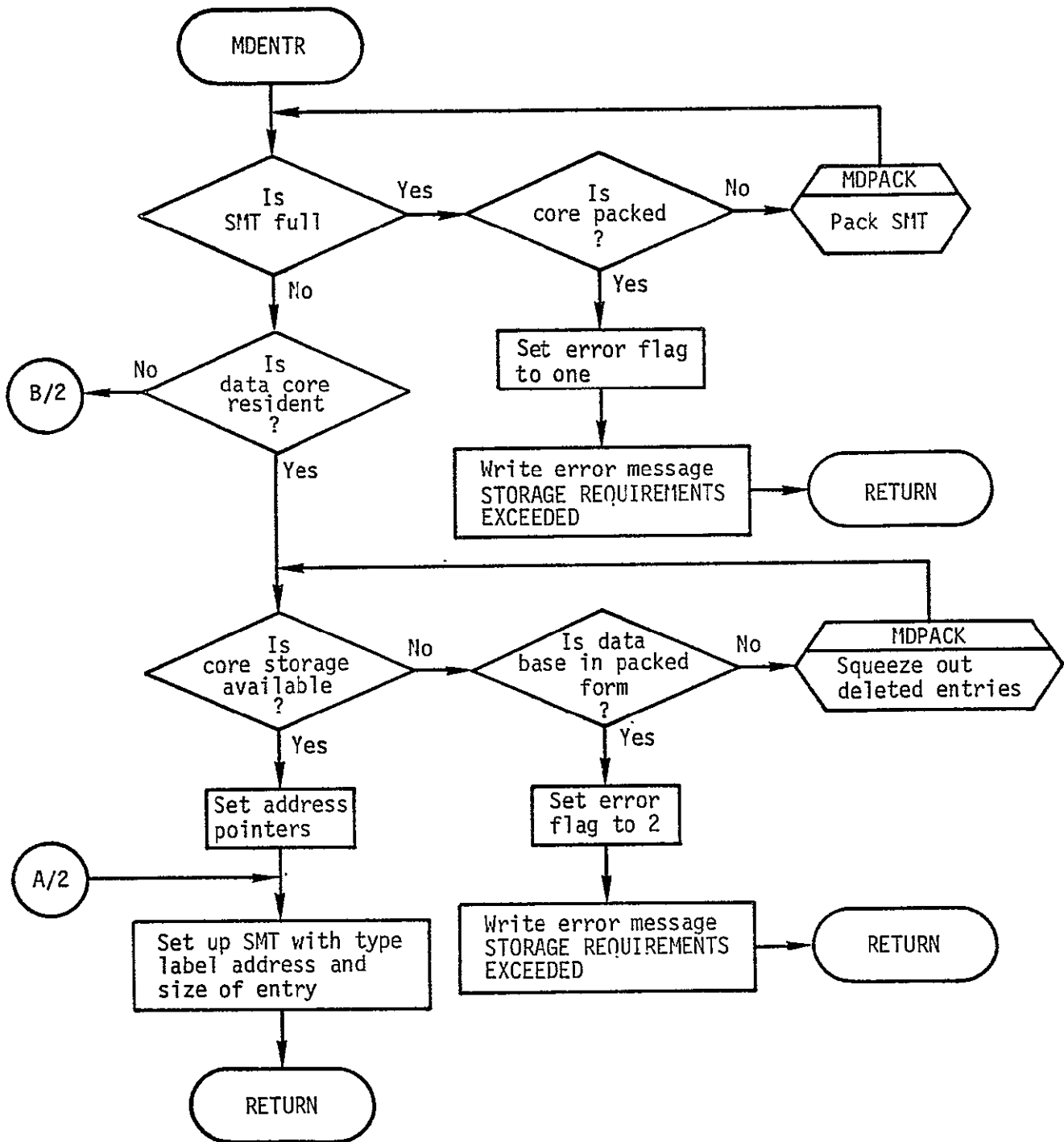
EXTERNAL STORAGE
NONE

BLANK COMMON
VARB I/O

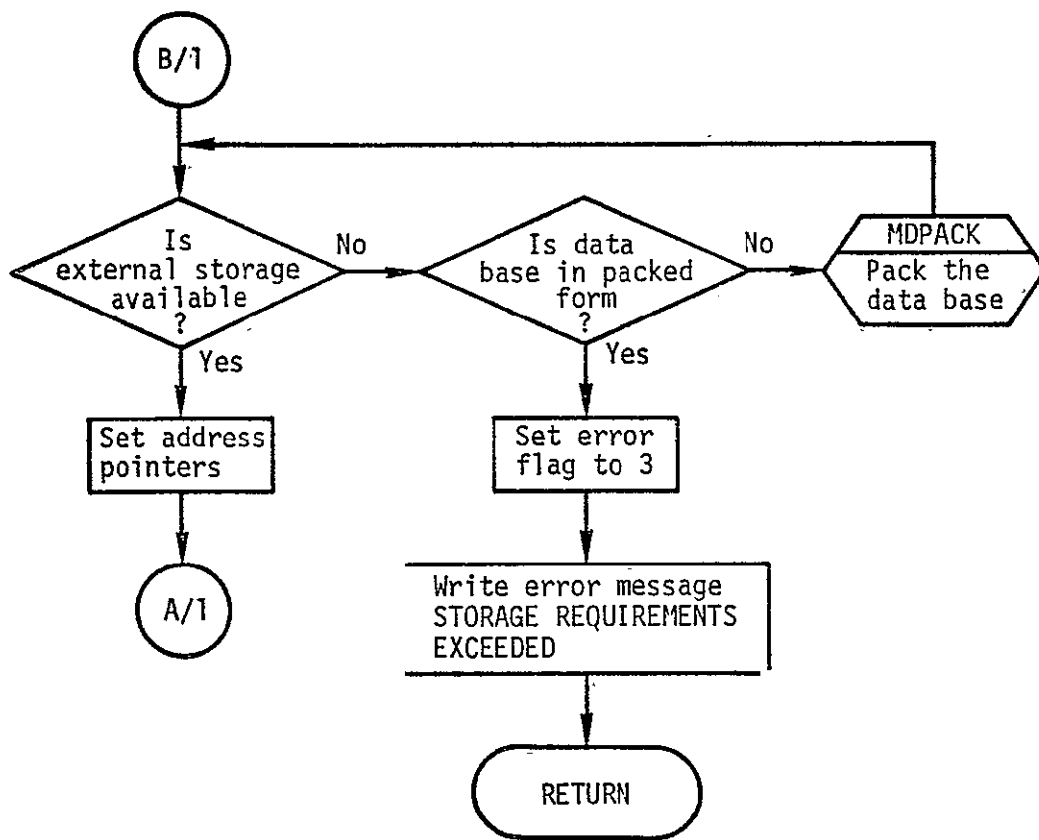
DBADDR I/O
EXADDR I/O
EXMAX I
NTRY I/O
PACK I
SORTFG O

LOCAL COMMON *
NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDENTR Flow Diagram



MDFIND - Storage Monitor

MDFIND is used to locate an entry in the storage monitor table (SMT).

Method

Input: The inputs to MDFIND are the label and type of the desired SMT entry. If no search is to be made on type, that input will be negative.

Processing: SORT1 is called to do an alphabetic sort on labels in the SMT and, if desired, to sort on type also. This sort is done only if the SMT is not already sorted in the above way. The SMT is then searched for the input name and type, if type was input. If the entry in the SMT was found, the output flag is set to one and the routine returns. The size of data, entry type of SMT, address of the data and the address of the SMT entry are output along with the output flag set to zero. The routine then returns to the calling routine.

Output: The outputs from MDFIND are the entry type, the size and address of the data and the address of the SMT entry. A flag is also output to indicate whether the desired entry was found or not.

USAGE

ENTRY MDFIND

CALL MDFIND (LABEL,TYPE,SIZE,ADDR,N,NOFIND)

ARGMT	I/O	TYPE	DIM	DEFINITION
LABEL	I	I	1	LABEL OF THE DESIRED SMT ENTRY ON INPUT TYPE IS THE ENTRY TYPE. IF NEGATIVE NO SEARCH FOR TYPE IS MADE. ON OUTPUT TYPE IS THE ENTRY TYPE FROM THE SMT
TYPE	I/O	I	1	
SIZE	O	I	1	SIZE OF DATA - IF NEGATIVE, DATA RESIDES ON EXTERNAL STORAGE DEVICE
ADDR	O	I	1	ADDRESS OF DATA - IF DATA IS CORE RESIDENT ADDR IS RELATIVE TO BLANK COMMON. IF DATA RESIDES ON EXTERNAL STORAGE DEVICE, THE NEGATIVE ADDRESS IS RETURNED
N	O	I	1	ADDRESS OF THE SMT ENTRY RELATIVE TO BLANK COMMON
NOFIND	O	I	1	FIND FLAG =0 DESIRED ENTRY WAS FOUND IN THE SMT =1 ENTRY WAS NOT FOUND IN THE SMT

EXTERNAL REFERENCES

SORT1
SEARCH

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

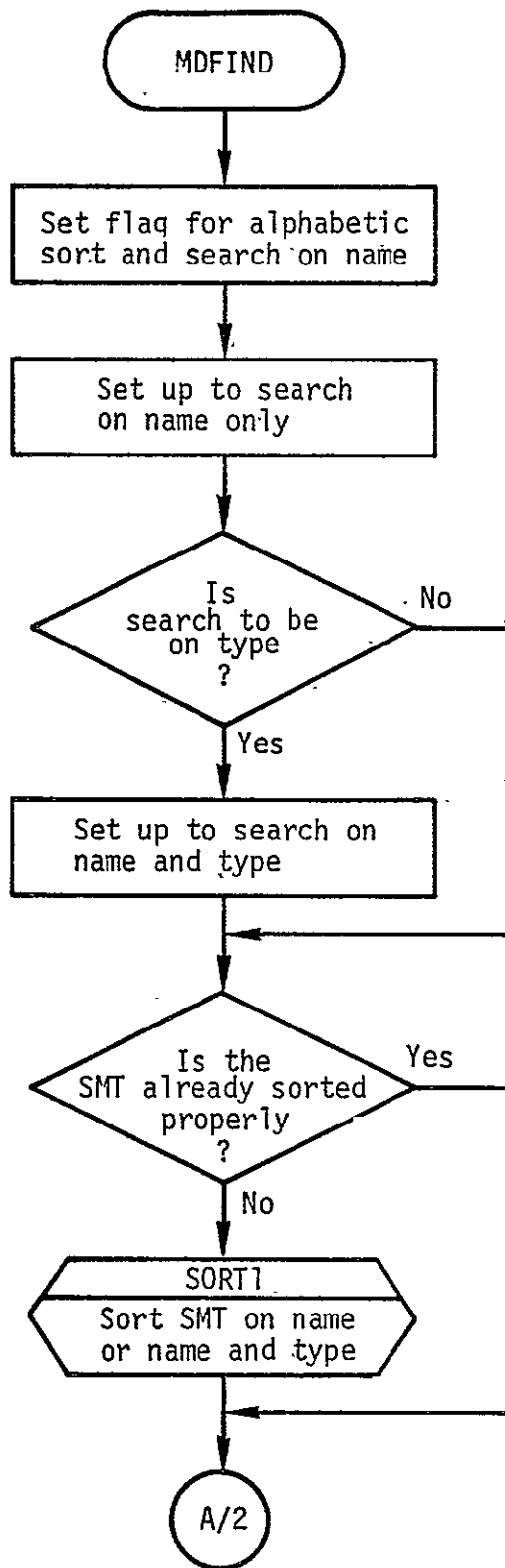
VARB I/O

DBSTRT I
NTRY I
SORTFG I

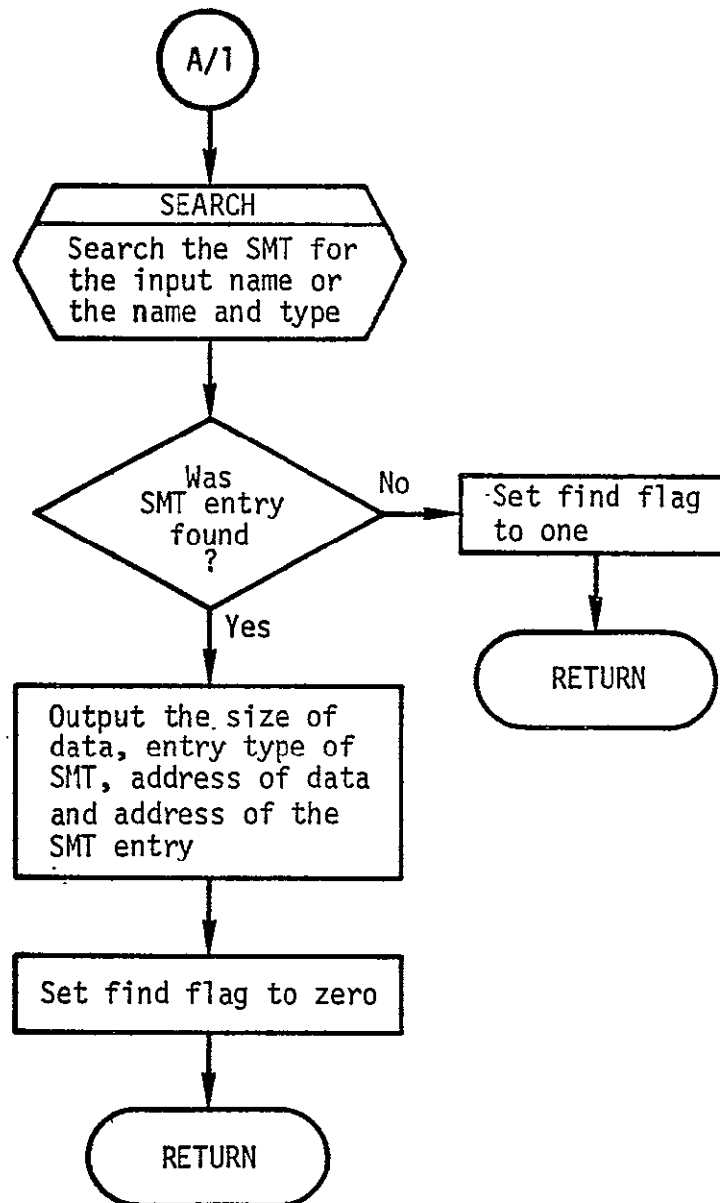
LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDFIND Flow Diagram



MDGET - Storage Monitor

MDGET retrieves data from the storage monitor table (SMT) and stores the data in an output buffer.

Method

Input: The inputs to MDGET are the name and entry type of the SMT entry where the data resides. The Ith and Jth location within the name to begin the retrieval must be input along with the maximum size of the output buffer.

Processing: The SMT is searched to find the proper entry in the SMT. If the entry was not found the status flag is set to -1 and the routine returns. The displacement within the data as specified on input and the size of the data is calculated. If the size of the data is not the same as the maximum, the status flag is set to 1 and the routine continues. If the calculated size of the data is greater than the input maximum size, the output size is set to the maximum size. If the data is in memory, the data is moved into the output buffer and the routine returns. If the data is to be on RAD, MDRADI is called to store the data on RAD. If an error occurred on the RAD store, the status flag is set to -3 and the routine returns. (Currently an attempt to store data on RAD will result in a termination of execution.)

Output: A buffer containing the desired data is output from MDGET along with the SMT entry type, number of words in the buffer and a status flag.

JSAGE

ENTRY MDGET

CALL MDGET (NAME,TYPE,IDIS,JDIS,MAX,BUFF,SIZE,STATUS)

ARGMT	I/O	TYPE	DIM	DEFINITION
NAME	I	I	I	NAME TO BE FOUND IN THE SMT DIRECTORY ON INPUT TYPE IS THE SMT TYPE TO BE MATCHED IN SMT SEARCH. IF NEGATIVE TYPE IS NOT COMPARED ON OUTPUT TYPE IS THE TYPE AS FOUND IN THE SMT ENTRY
TYPE	I/O	I	I	
IDIS	I	I	I	DISPLACEMENT FOR THE I-DIMENSION OF NAME
JDIS	I	I	I	DISPLACEMENT FOR THE J-DIMENSION OF NAME
MAX	I	I	I	MAXIMUM SIZE OF OUTPUT BUFFER
BUFF	O	I	SIZE	DATA FROM NAME(IDIS,JDIS)
SIZE	O	I	I	NUMBER OF WORDS MOVED INTO 5466
STATUS	O	I	I	STATUS FLAG = 1 NUMBER OF WORDS TRANSFERED WAS NOT EQUAL TO MAX = 0 DATA TRANSFERED OK =-1 NAME WAS NOT FOUND =-3 ERROR OCCURED IN ATTEMPT TO READ DATA FROM RAD

EXTERNAL REFERENCES

MDFIND
MDRADI

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

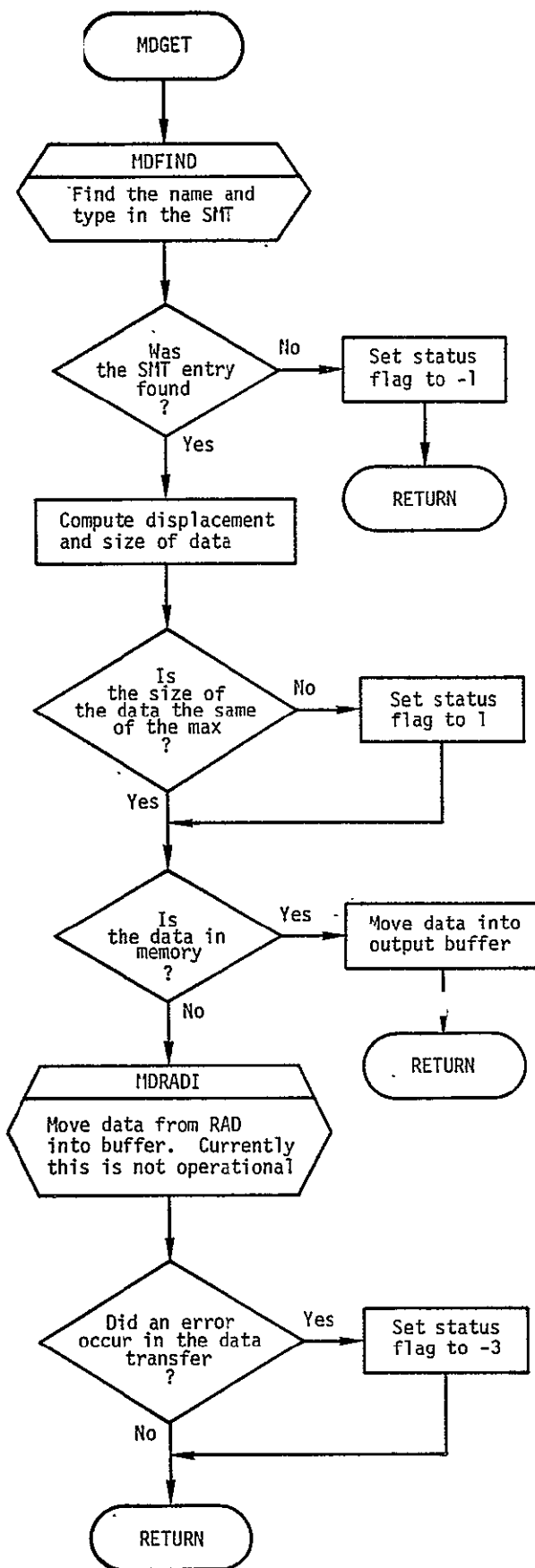
BLANK COMMON

NONE

LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDGET Flow Diagram

MDPACK - Storage Monitor

MDPACK removes deleted entries from the storage monitor table (SMT) and packs the data area.

Method

Input: The storage monitor table with the entries to be removed flagged for deletion and the SMT accounting information are input to MDPACK along with the device (memory or RAD) to pack.

Processing: For a memory pack MDPACK locates the entries flagged for deletion in the SMT, set the deletion flag (TYPE = -1) to a large number, and shifts the remaining data base squeezing out the data for the deleted entry. The address portion of the SMT is updated simultaneously. Packing of the SMT is accomplished by sorting the SMT by type, i.e., moving the deletions to the bottom and resetting the number of SMT entries.

The mechanism of the above procedure is to check each entry in the SMT from the last entry to the first. Each entry is noted as to whether or not the entry is to be deleted. The data base is squeezed after there has been detected a deleted entry after a non-deleted entry, excluding the first deleted entry. The data base is also squeezed after all entries are checked, if it needs to be. The PHAZ flag keeps a record of the entries. The definition of PHAZ is

- = 0 no deleted entries found yet
- = 2 last entry was deleted
- = 3 last entry was not deleted
- = 4 end of SMT, last entry not deleted and the final data base squeeze has not occurred

For a RAD pack, MDPACK returns.

Output: The storage monitor table with the deleted entries removed and the packed data area are output from MDPACK.

ISAGE

ENTRY MDPACK
CALL MDPACK (DEVICE)

ARGMT I/O TYPE DIM

DEVICE I I I

DEFINITION

DEVICE TO BE PURGED ALONG WITH THE
SMT
#0 PACK USER CORE
#1 PACK USER DATA ON EXTERNAL STORAGE
DEVICE

EXTERNAL REFERENCES

SORT1

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

VARB I/O

DBSTRT I

NTRY I

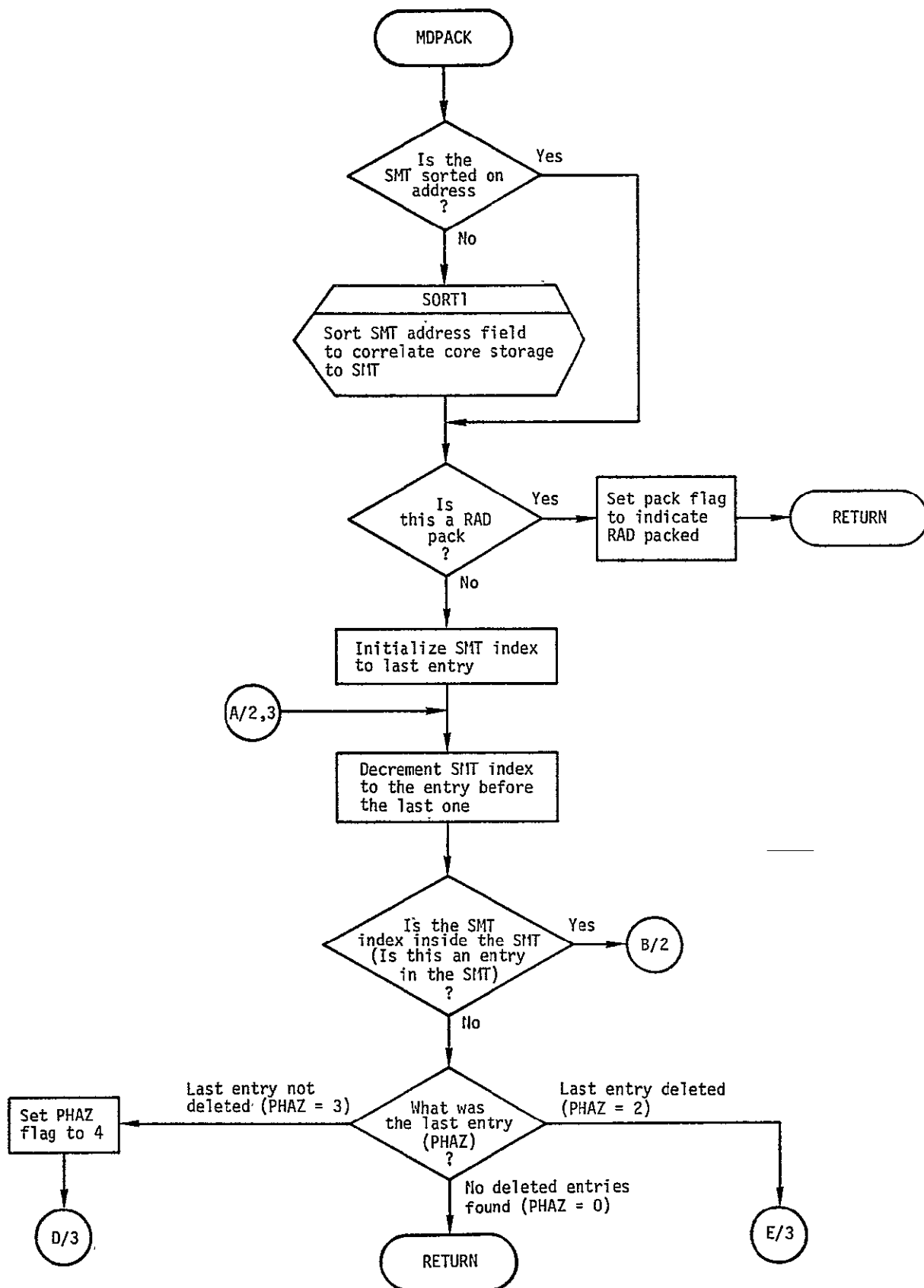
PACK 0

SORTFG I/O

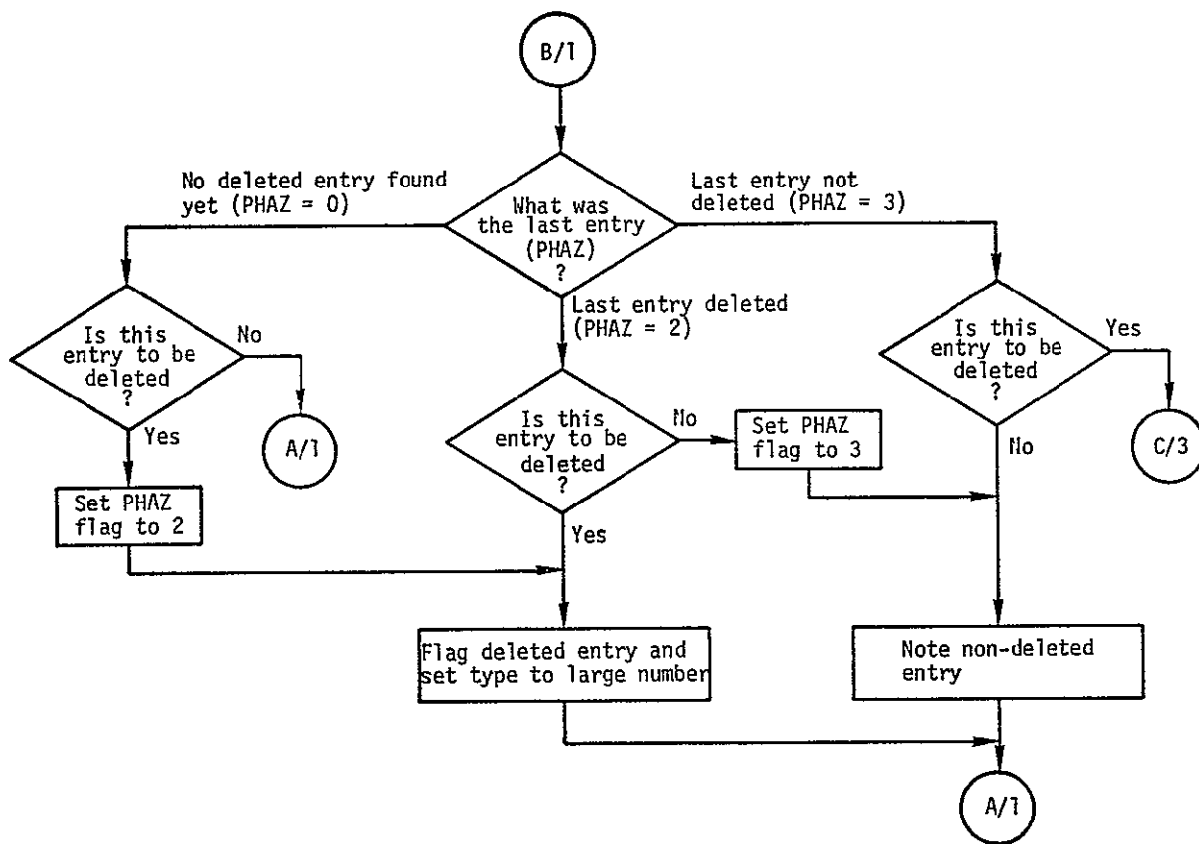
DBADDR 0

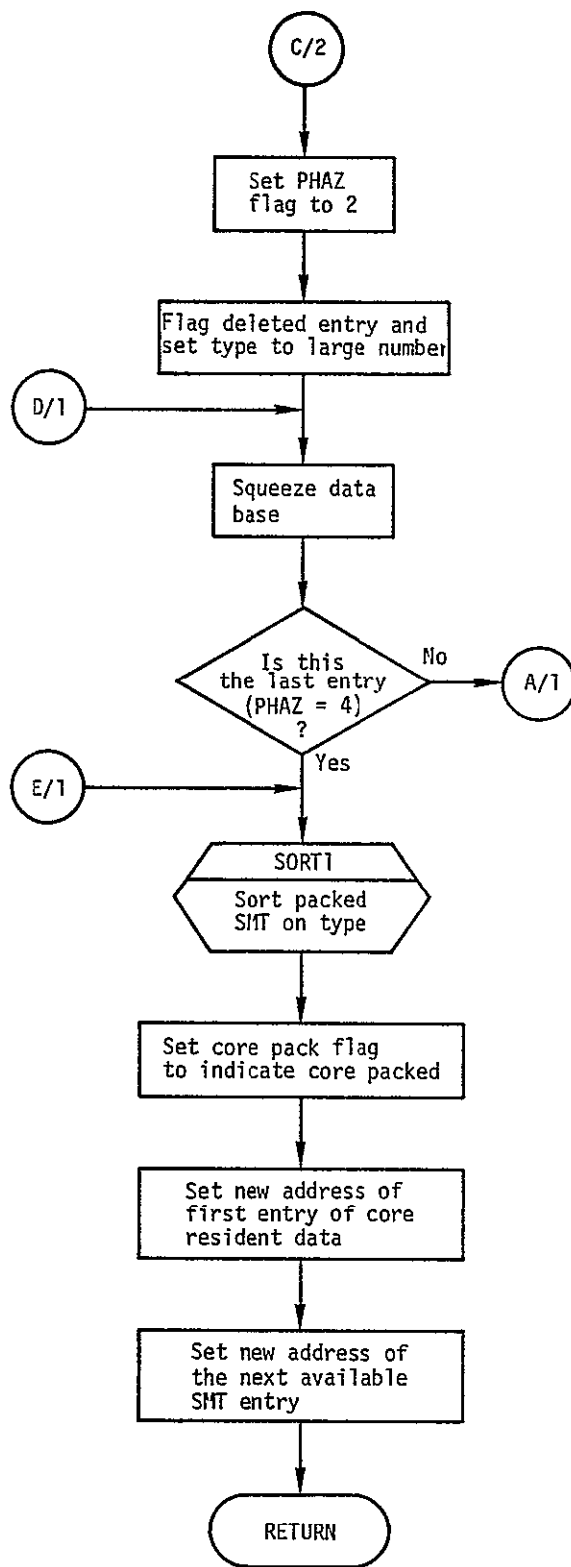
LOCAL COMMON

NONE



MDPACK Flow Diagram





MDPUT - Storage Monitor

MDPUT puts data into the storage monitor table (SMT).

Method

Input: A description of the SMT entry of where to place the data is input. The data along with a flag to indicate where the data is to be placed (memory or RAD) is also input.

Processing: The SMT is searched for the proper entry. If an entry was found for the data name and the data in the SMT does not have the same attributes as the input data (i.e., resides on the same device and has the same length) the entry in the SMT is deleted. If an entry was not found or was deleted, storage will be allocated in the SMT for that entry. If an error occurs in the allocation, the status flag is set to -1 and the routine returns.

If the device to place the data is memory, the data is moved into memory, the status flag set to 0 and the routine returns. If the device is external storage, MDRAR0 is called to store the data on RAD. If an error occurs in the RAD store, the status flag is set to -3. MDPUT then returns to the calling routine.

Output: Data is placed in the SMT with a new entry in the SMT if one did not exist for the desired data name. A status flag is also output indicating if an error had occurred.

USAGE

ENTRY MDPUT

CALL MDPUT (NAME,TYPE,LENGTH,IDIM,BUFF,DEV,STATUS)

ARGMT.	I/O	TYPE	DIM	DEFINITION
NAME	I	I	I	NAME OF THE SMT ENTRY IN WHICH TO PUT THE DATA
TYPE	I	I	I	DATA TYPE FLAG
LENGTH	I	I	I	SIZE OF THE SMT ENTRY
IDIM	I	I	I	COLUMN DIMENSION OF ENTRY
BUFF	I	I	LENGTH	DATA BUFFER
DEV	I	I	I	MEMORY/RAD FLAG =0 MEMORY =1 RAD
STATUS	O	I	I	STATUS FLAG = 0 OK =-1 COULD NOT ENTER SMT ENTRY =-3 RAD WRITE FAILED

EXTERNAL REFERENCES

MDFIND

MDELET

MDENTR

MDRADO

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

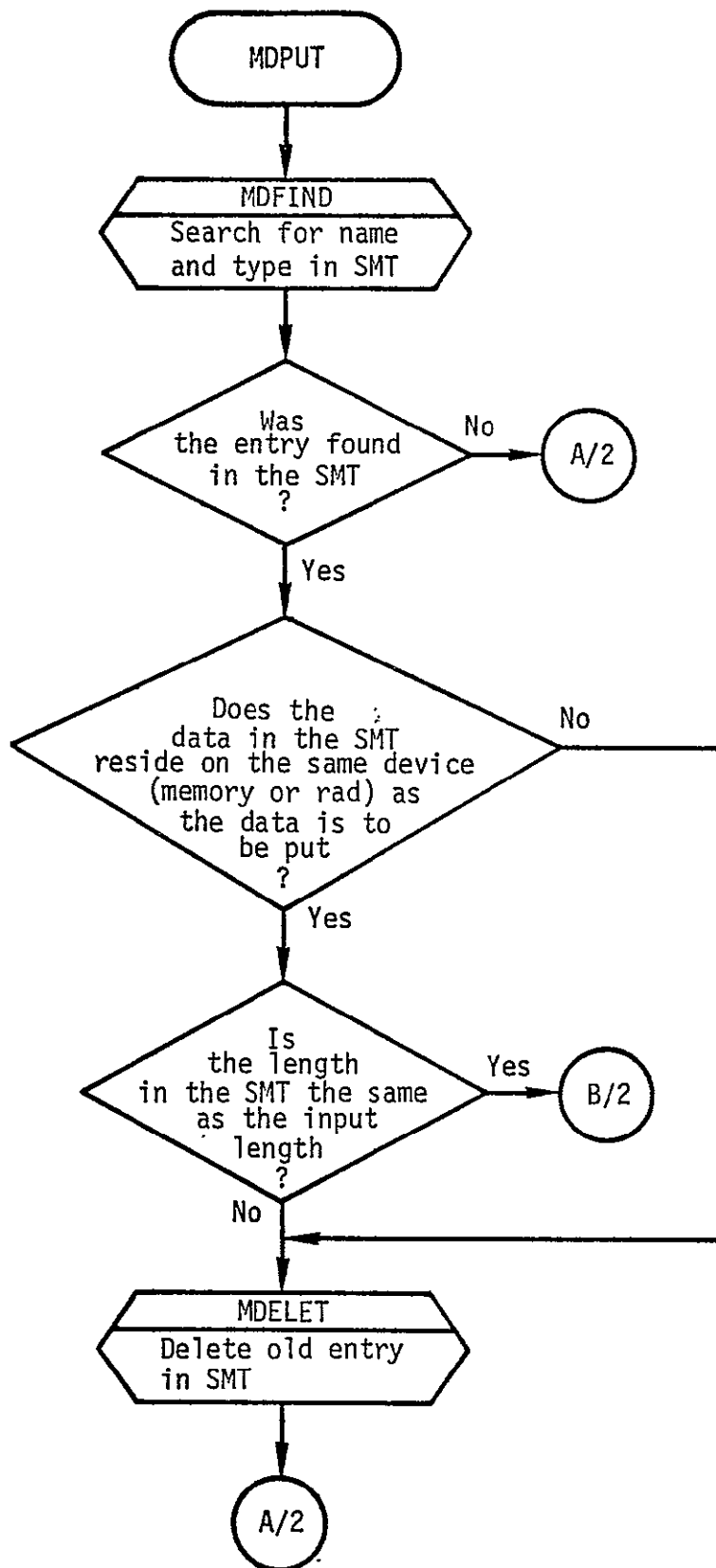
BLANK COMMON

NONE

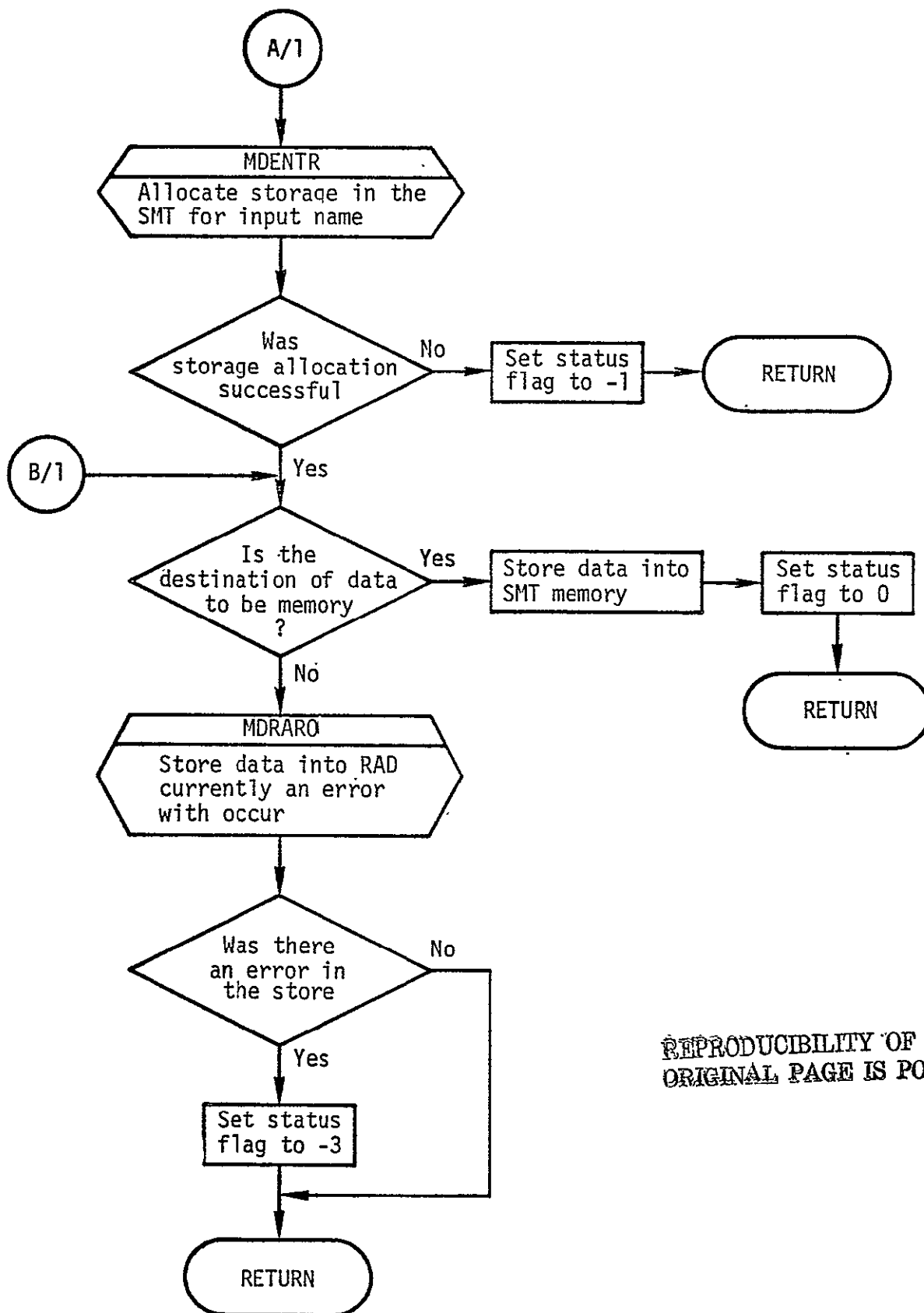
LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDPUT Flow Diagram



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDRADI - Storage Monitor

MDRADI, when developed and implemented, will retrieve data from the RAD (random access device) portion of the SMT and place it into the provided memory buffer. MDRADI is the mechanism for acquiring particular data elements which are in the RAD portion of the SMT, whereas MDROLL is the mechanism for transferring data between the memory and RAD portions of the SMT.

MDROLL - Storage Monitor

MDROLL's purpose is to bring into memory all data required by a processor for execution and, if necessary, will roll data not required onto RAD. However, currently the RAD is not defined therefore MDROLL does not perform the above function but exists to provide the interface. Currently MDROLL only determines if enough memory is available.

Method

Input: The number of data words required by a processor to execute is input to MDROLL.

Processing: When the number of words available is less than the words required for execution, additional code must be implemented to place data on RAD. This code will determine what is necessary to remain in memory, determine the hierarchy of data to go to RAD and will write this data on RAD. The logic of the current MDROLL is shown in the figure with a comment where the proposed code should be inserted.

Output: A status flag indicating the availability of memory is output. The proposed output will be data on RAD that will not fit into memory..

USAGE

ENTRY MDROLL
CALL MDROLL (NWORDS,STATUS)

ARGMT	I/O	TYPE	DIM	DEFINITION
NWORDS	I	I	1	NUMBER OF WORDS IN SMT REQUIRED
STATUS	O	I	1	STATUS FLAG
				* 0 NUMBER OF WORDS AVAILABLE IS GREATER THAN THAT REQUIRED
				* -1 NUMBER OF WORDS AVAILABLE IS LESS THAN THAT REQUIRED

EXTERNAL REFERENCES
MDPACK

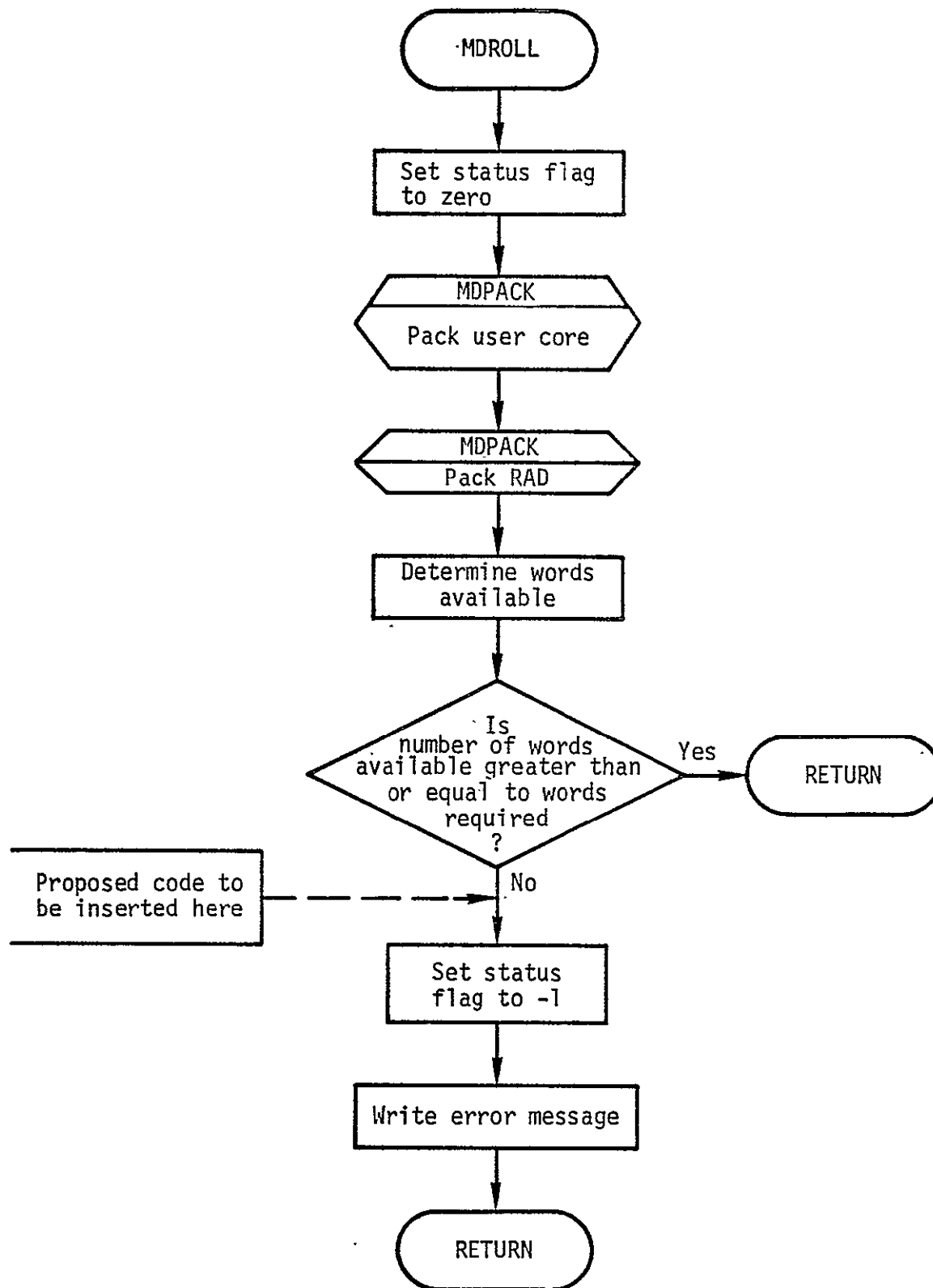
DIAGNOSTICS
MEMORY REQUIREMENTS EXCEEDED
..... WORDS REQUIRED WORDS AVAILABLE
NUMBER OF WORDS REQUIRED IS GREATER THAN THAT
AVAILABLE. STATUS FLAG WILL BE SET TO -1.

EXTERNAL STORAGE
NONE

BLANK COMMON
VARB I/O

DBADDR I
NTRY I

LOCAL COMMON
NONE



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDROLL Flow Diagram

MDALOC - Execution Controller

MDALOC establishes the input and output arguments' linkages for a processor, allocates storage for all output parameters, and communicates parameters scan control data to the resident.

Method

Input: The control table, in the working buffer, contains the argument specifications and data input to MDALOC.

Processing: MDALOC is entered from MDSMON for each command to establish the argument linkage. MDALOC's function is to allocate storage for output variables, to determine argument addresses of input and output variables and to initialize scan values.

The control table containing the argument specification is passed to MDALOC through a working buffer in common. If the control table is incomplete, the status flag is set to -1 and MDALOC returns; otherwise, the control table is packed before proceeding.

Before the allocation of storage for output variables can be performed, MDALOC must determine the amount of memory not yet allocated in order to decide if the total amount of memory will be more than the available memory. This is accomplished by first initializing a memory counter to the known memory which includes the memory needed for immediate data (=). Each argument is examined. If the argument is immediate data, no memory is added to the memory counter. If the argument data is indirect (@), memory will be added to the memory counter, ignored or an error will occur depending on the argument's input/output. For output arguments the memory counter will be incremented by the size of the data. If an SMT entry already exists for this name, the type of the output array to be created and the SMT type that exists must be the same and the size of the SMT entry must be larger than the array to be created or the old SMT entry will be deleted. If input arguments are not immediate data or in IMS, an error will occur. In no case will memory be added to the counter for an input argument. After all arguments are checked MDROLL is called to determine if enough memory is available and if not to roll part of data to RAD (not implemented). The new control table is then placed in the SMT with the name &CONTB and with a type of 1000. A detailed description of above is depicted on pages 1 - 5 of the flow chart.

The allocation of storage for output arguments and the setting of the argument address are shown in the flow chart on pages 5 - 7. For each argument the address within the data base that contains the data is calculated and the address is placed in blank common (variable name ARGADD dimensioned 30). If an argument is not found in the SMT with the same name and type, MDENTR is called to allocate storage and to build an SMT entry for that argument. If an argument is incomplete and is not a scan variable, the completion flag for the entire control table is set to incomplete, an error message is printed, and MDALOC returns. The argument data address for immediate data is simply the relative address within the control table plus the address of the control table. For indirect data, the address is calculated by determining the amount of memory left in the control table and subtracting that from the next available cell in memory after the control table. The addresses of the scan variables are set to B(36) and B(41) in blank common.

The scan initialization performed by MDALOC is shown in pages 8 - 10 of the flow chart. If the scan is activated, the preamble table of the data box is set up. The definition of the table follows:

format flag	number of dependent variables in summary vector
name of X variable	name of Y variable
units of X variable	units of Y variable
X centroid of scan	Y centroid of scan
X step size	Y step size
number of steps to be taken on each side of X centroid	number of steps to be taken on each side of Y centroid

If any of the two scan arguments is incomplete, the values for that argument are zeroed. The scan variables within the intramonitor communication data area are also set. The preamble table is output to RAD and the file opened. The argument text definition text is then output to the same file. Currently the argument text definitions are blank. The data box is now initialized and ready to accept data.

Output: The addresses of argument data are placed in blank common and the control table from the working buffer is placed in the SMT with the name &CONTB and type of 1000. If a scan is activated for this processor, the scan values are placed in blank common and the preamble to the data box is output to RAD.

AGE

ENTRY MDALOC
CALL MDALOC (STATUS)

ARGMT I/O TYPE DIM

DEFINITION

STATUS 0 I 1

STATUS FLAG

- 0 STATUS OK
- -1 CONTROL TABLE INCOMPLETE
- -2 CONTROL TABLE DATA NOT FOUND
- -3 UNDEFINED INPUT ARGUMENT
- -4 DATA IS ON RAD/NOT IMPLEMENTED
- -5 ERROR IN MDENTR
- -6 ERROR IN MDPUT
- -7 ERROR IN SCAN DATA
- -8 ERROR IN MDPUTC
- -9 I/O ARG. OF INSUFFICIENT SIZE

EXTERNAL REFERENCES

MDCTPK
MDSPEC
MDLKUP
MDFIND
MDIMS1
MDELET
MDROLL
MDPUT
MDENTR
MDCNTS
MDPUTC

DIAGNOSTICS

***** [TYPE] DELETED.

THIS ARGUMENT IS OUTPUT DATA THAT IS NOT IMMEDIATE AND
THE TYPES DO NOT MATCH THEREFORE THE OLD SMT ENTRY IS
DELETED.

***ERROR ENCOUNTERED WHILE PREPARING FOR
PROCESSOR EXECUTION.

STATUS FLAG HAS BEEN SET TO A NEGATIVE NUMBER.

SEE DEFINITION OF THE FLAG FOR POSSIBLE ERROR.

EXTERNAL STORAGE

THE PREAMBLE AND THE ARGUMENT DEFINITION TEXT OF THE
DATA BOX IS OUTPUT TO RAD.

BLANK COMMON

VARB I/O

ARGAD 0
ARGADD 0
DBSVLN 0
NUMARG 0
SCANF 0
SCNVAL 0
VERSION 1

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

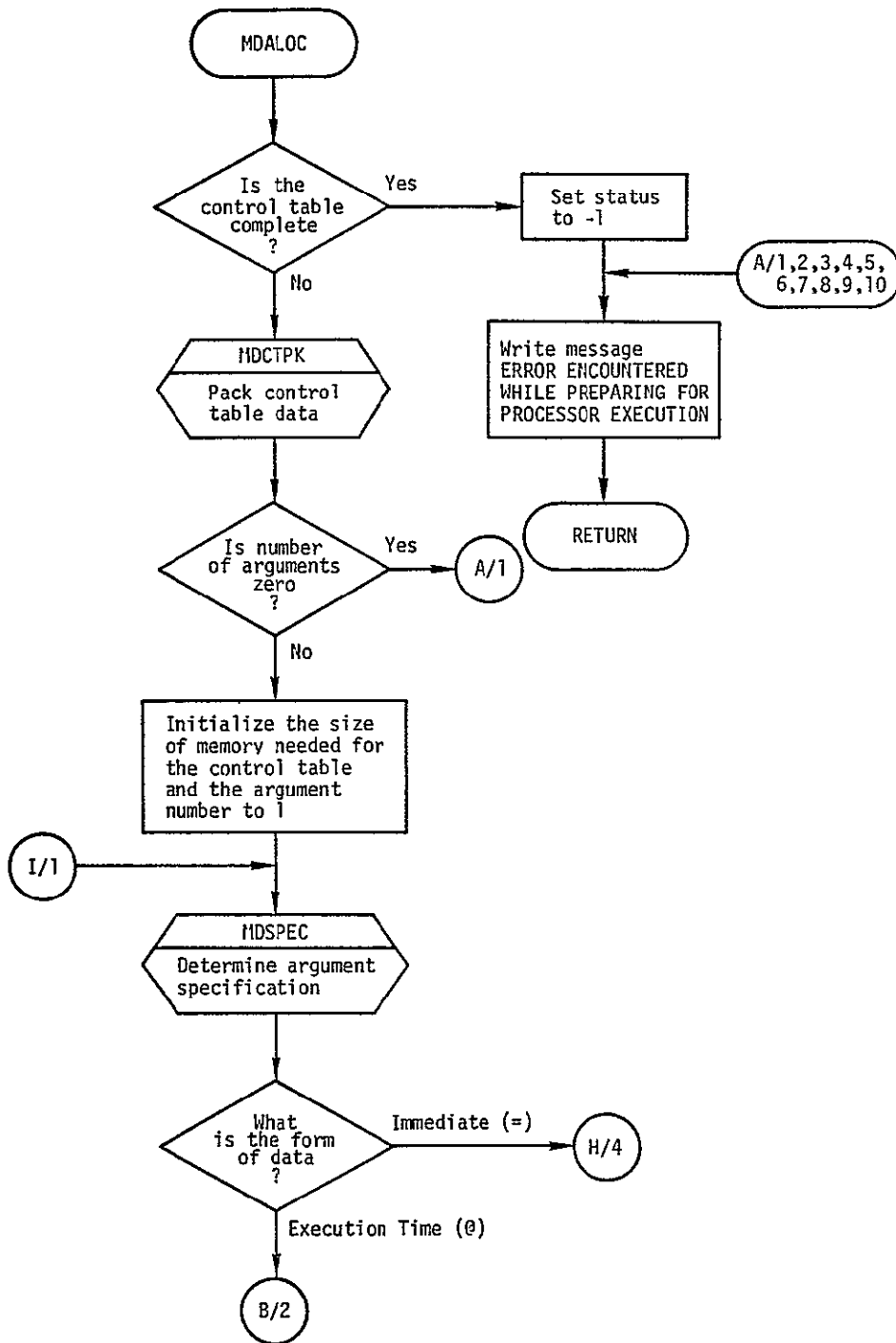
COMMON /MDBUFF/

VARB	I/O
-------------	------------

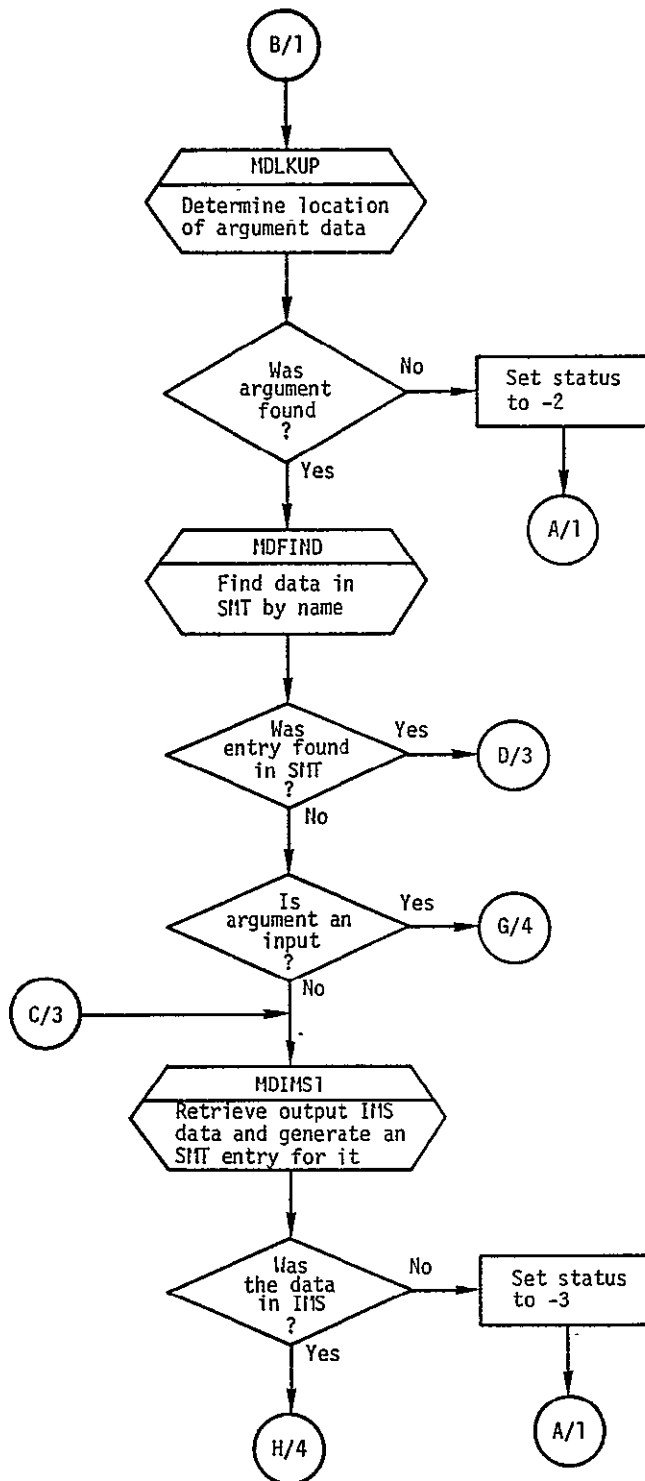
BDATA	I
--------------	----------

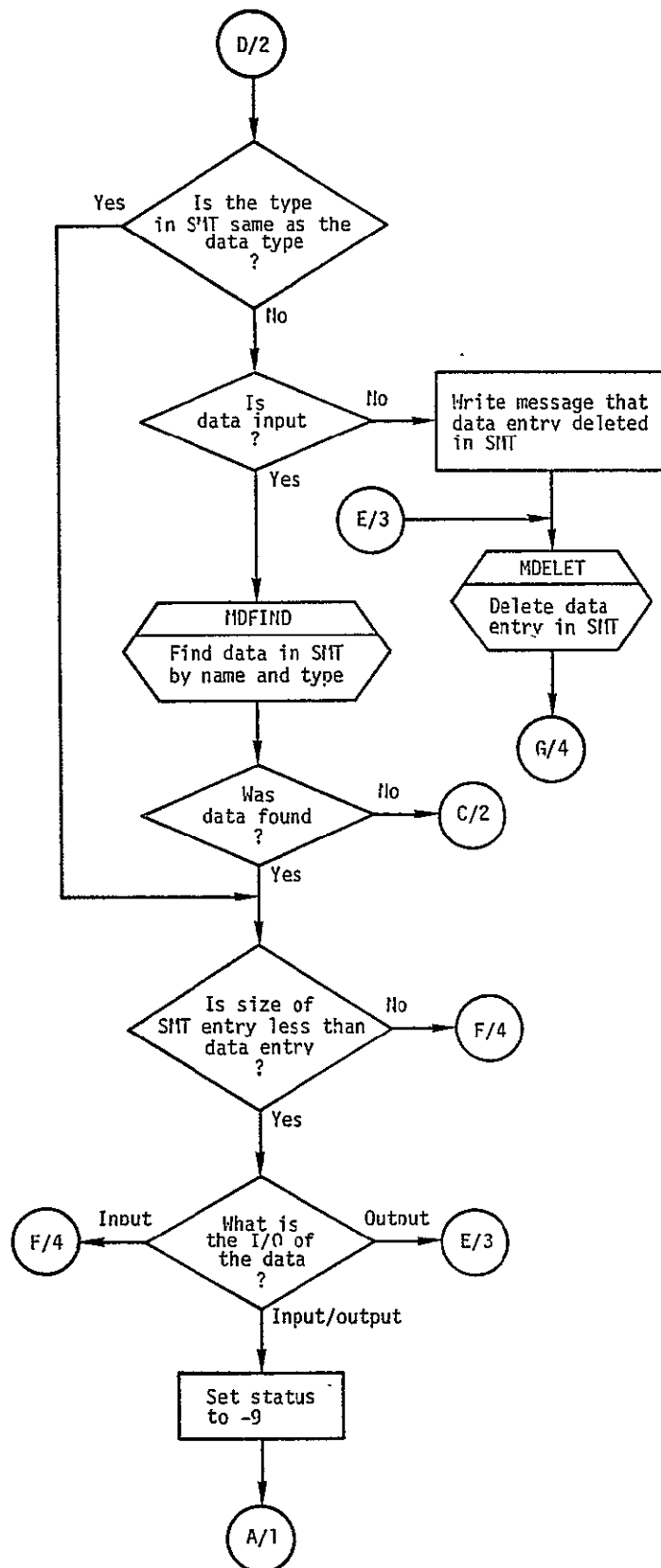
DSIZE	I
--------------	----------

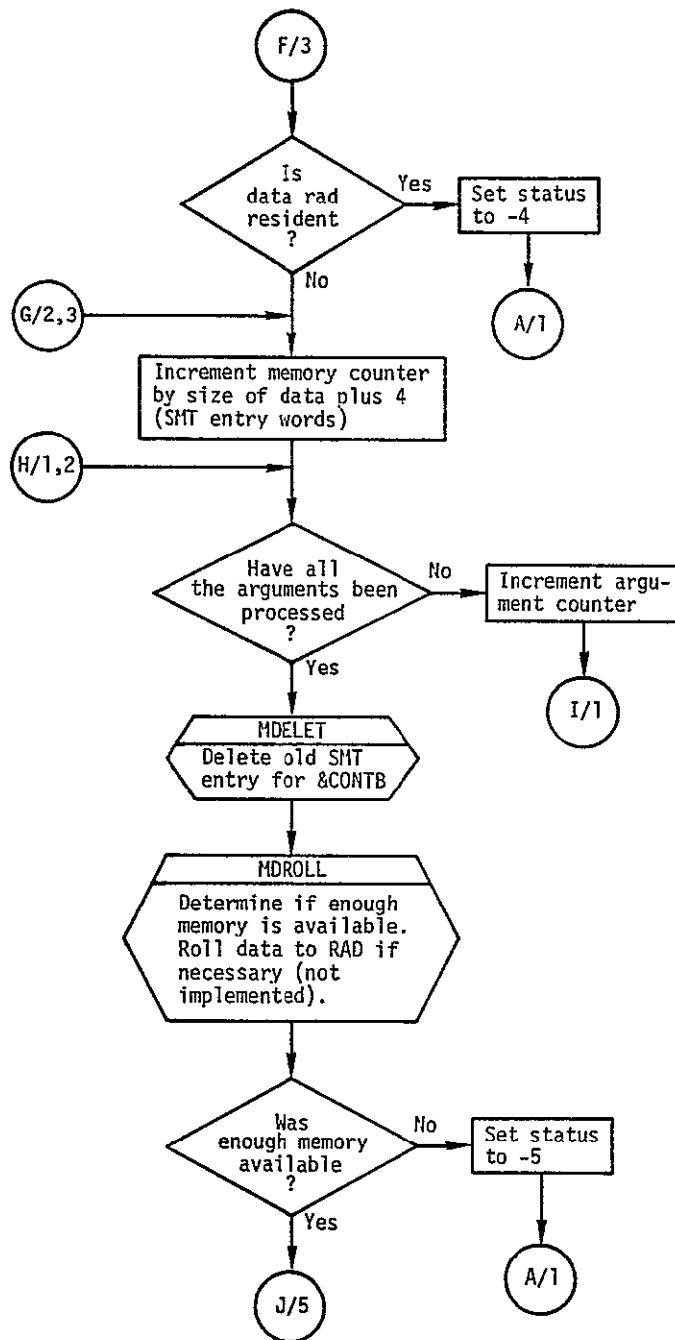
WBUF	I/O
-------------	------------



MDALOC Flow Diagram



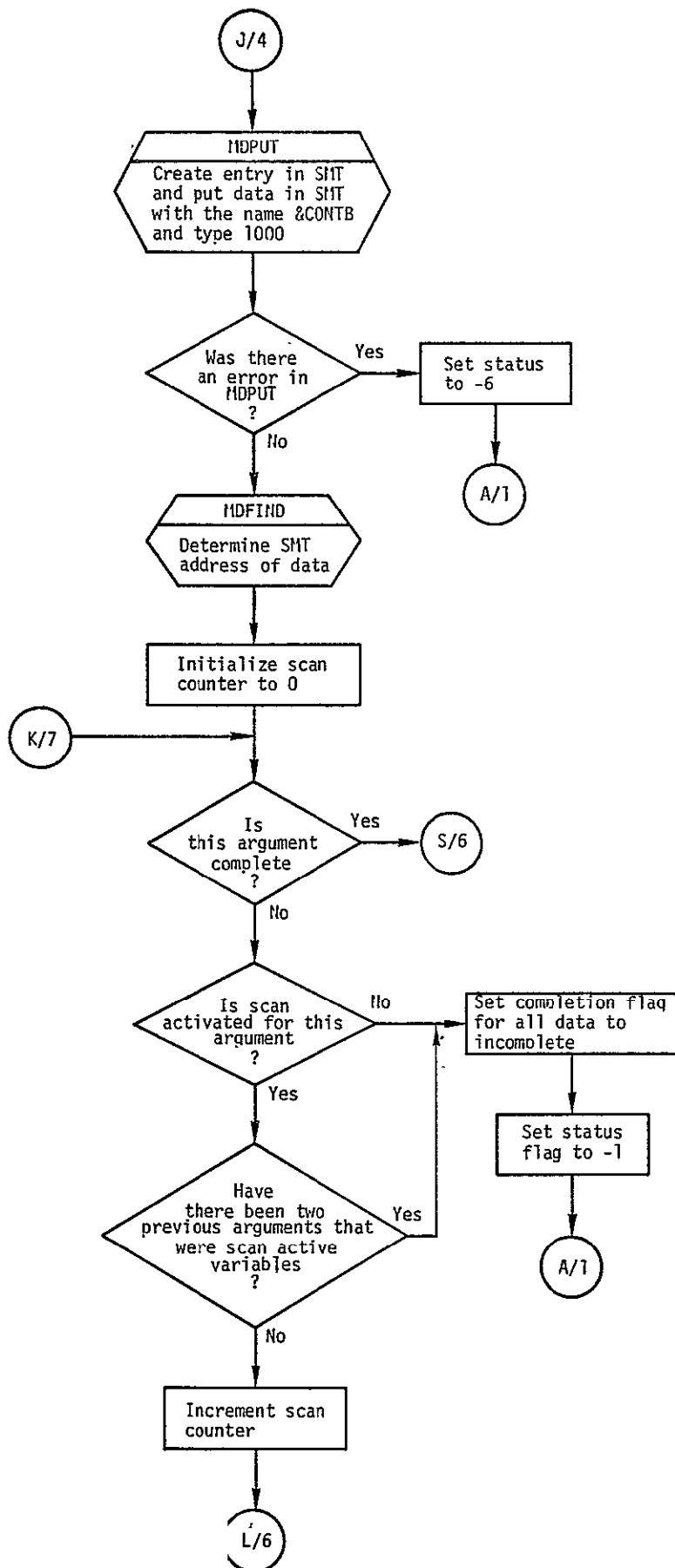




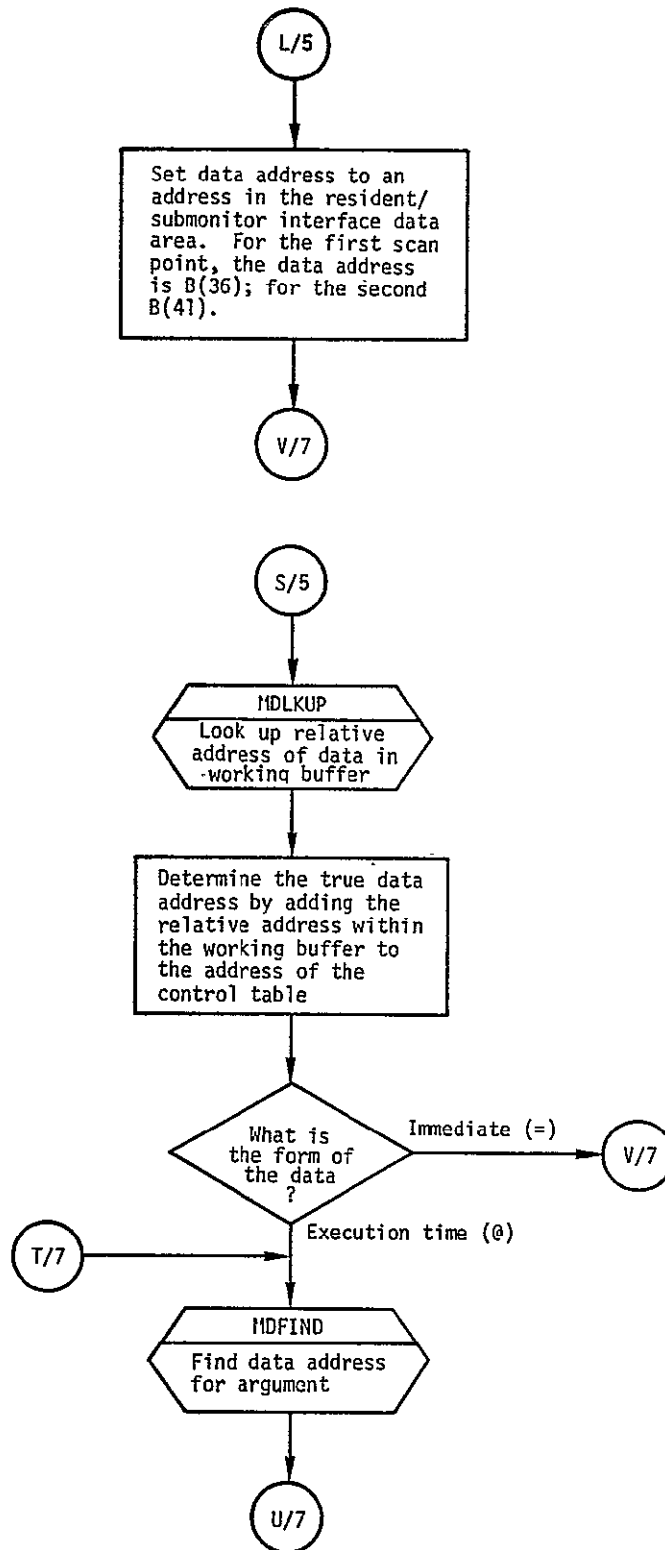
MDALOC Flow Diagram (Continued)

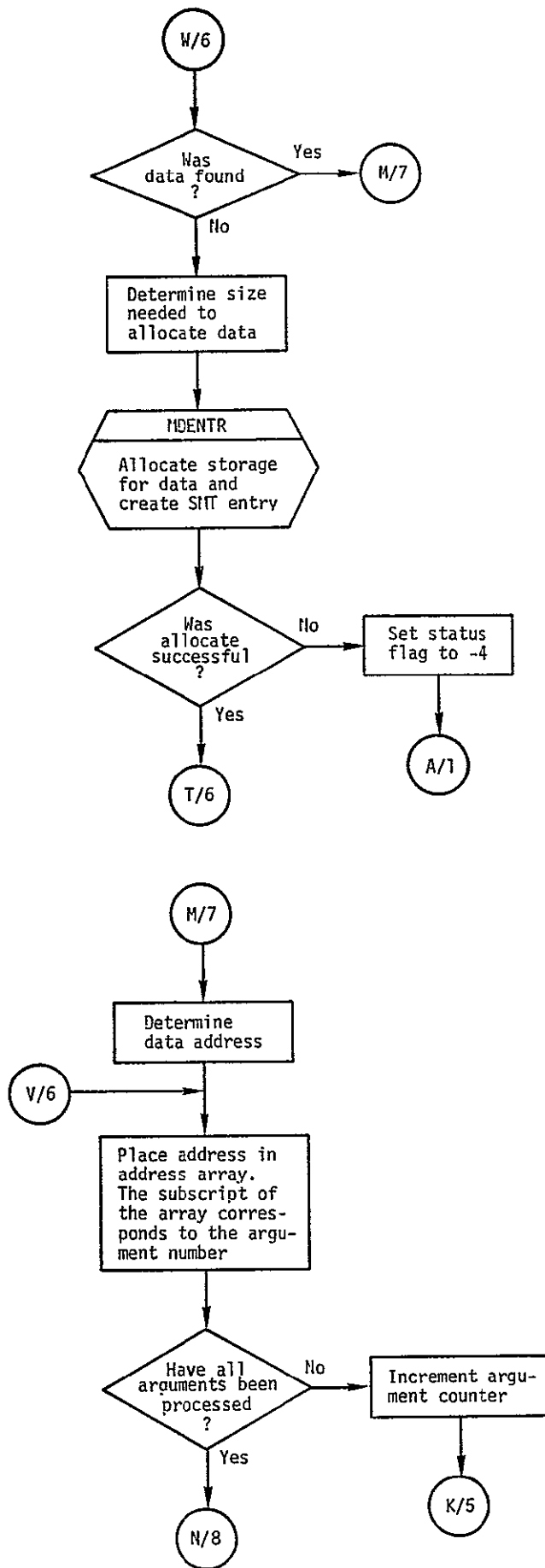
5.1-9

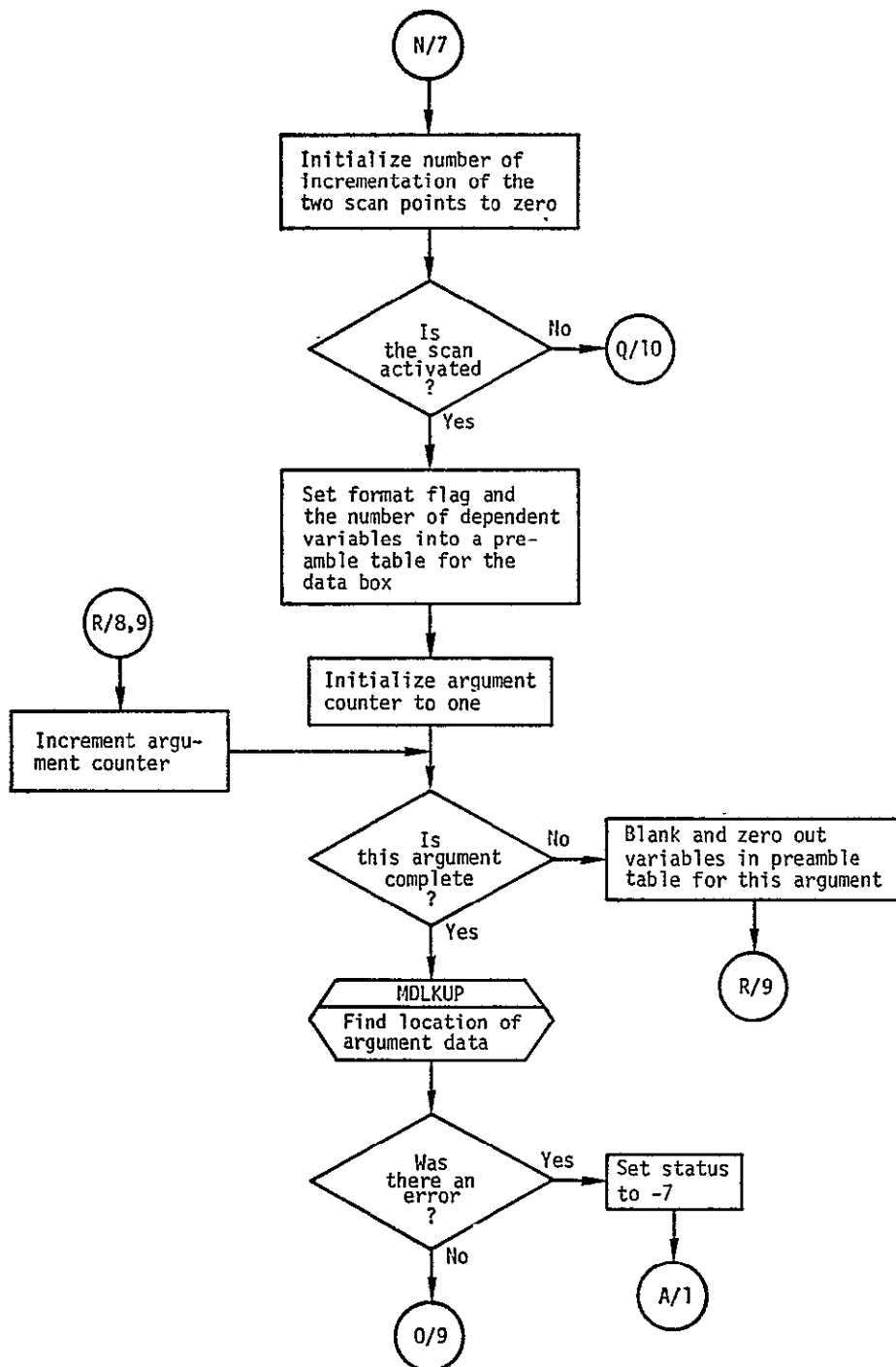
REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

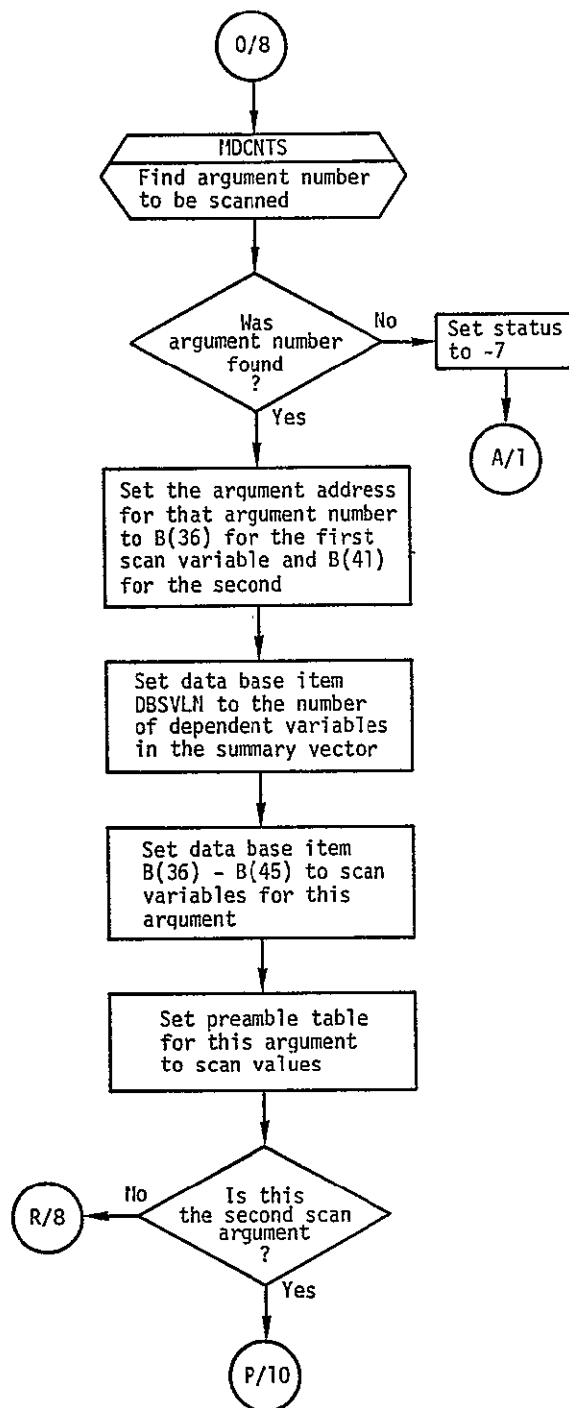


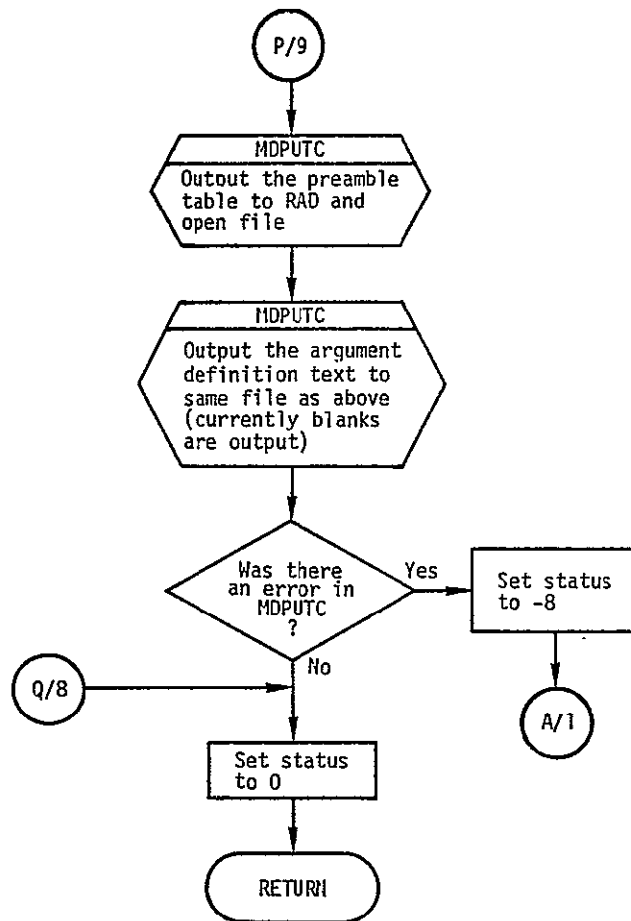
C-2











MDALOC Flow Diagram (Continued)

MDCMTG - Execution Controller

MDCMTG brings the command table into memory (SMT) and places the requested command into the intramonitor communications area of blank common.

Method

Input: The majority of the input is contained in blank common and is: the command table name, the command table type, the command number to be executed next and the number of the last command to be executed. The calling sequence contains a flag indicating if this is the first execution of MDCMTG for this command table. If it is, the command to be executed next and the last command are sequence numbers which are converted to command numbers.

Processing: The command table is brought into the memory portion of the SMT, if necessary. If this is the initial execution, the range of commands is checked for legality, and, if valid, converted to command numbers and stored in common. If an error exists in the request (i.e., bad sequence number) control is returned to MDSMON with an indication of this occurrence. An error message is also printed.

In each execution, the command number to be executed is compared to the last number to be executed and the last number is compared to the total number of commands in the table. If either comparison shows that the limits have been exceeded, an error message is output and control is returned to MDSMON with a status indicating such. If neither limit is exceeded, the command is broken into its characteristic parts and placed in common.

Output: A status indication is passed through the calling sequence. All other output is placed in blank common and is: a print flag, a temporary edit existence flag, the control table type, the control table name, the processor name and the sequence number of the command to be executed.

USAGE

ENTRY MDCMTG
CALL MDCMTG (STATUS)

ARGMT	I/O	TYPE	DIM
STATUS	I/O	I	I

DEFINITION

UPON ENTRY, STATUS INDICATES IF THIS IS THE INITIAL ENTRY: =0, INITIAL ENTRY, =1, NOT INITIAL ENTRY. UPON COMPLETION OF EXECUTION, STATUS INDICATES THE VALIDITY OF THE INPUT: =0, INPUT GOOD; =1, SYNTAX ERROR IN INPUT.

EXTERNAL REFERENCES

MDCMTS
MDFIND

DIAGNOSTICS

***NON-EXISTANT SEQUENCE NUMBER
THE USER HAS SPECIFIED A SEQUENCE NUMBER TO EXECUTE WHICH DOES NOT EXIST.

EXTERNAL STORAGE

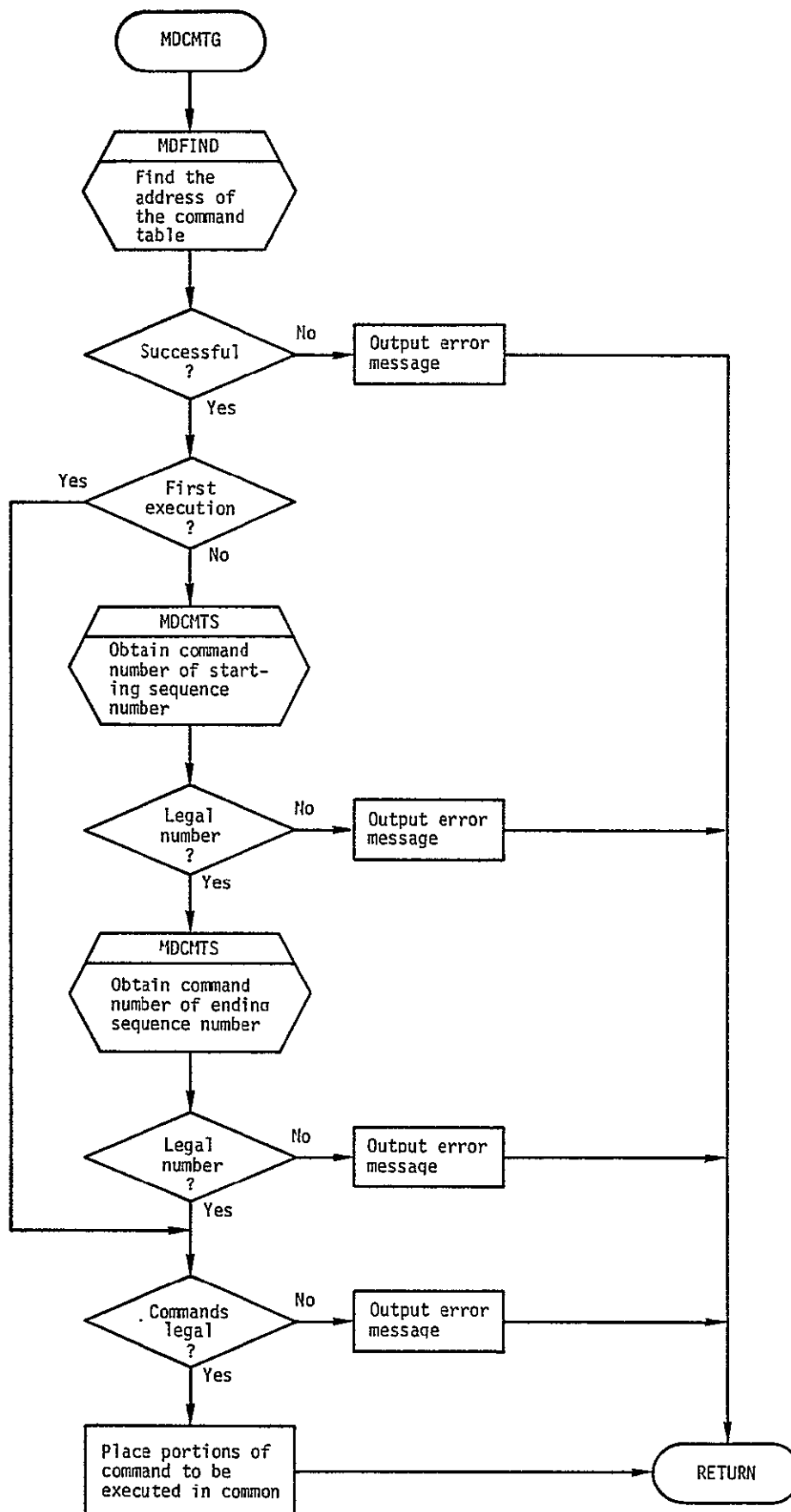
NONE

BLANK COMMON

VARB	I/O
CMDNO	I/O
CMTNAM	I
CMTYP	I
CTNAME	O
CTYPE	O
DIRECT	I
EDIT	O
ENDNO	I/O
ENTRY	I
PNAME	O
PRINT	O
SEQNO	O

LOCAL COMMON

NONE



MDCMTG Flow Diagram

MDCMTV - Execution Controller

MDCMTV is used to interpret and verify the directive given for SEMI, AUTO or AUTO* mode.

Method

Input: The only input to MDCMTV is a buffer containing the directive after processing by the user communications component.

Processing: After verifying the syntax of the directive, the command table type and name are extracted from the directive and placed in blank common. If the user has specified a range of commands to execute, the beginning and ending sequence numbers of this range are placed in common and control returned to MDSMON. If a range is not specified, zeros are placed in common in place of sequence numbers.

Output: A status flag indicating the validity of the directive is passed through the calling sequence. All other output is in blank common and is: the command table type, the command table name, the sequence number of the first command to be executed (or zero if not input) and the sequence number of the last command to be executed (or zero if not input).

USAGE

```
ENTRY MDCMTV
CALL MDCMTV(INPUT,STATUS)
```

ARGMT	I/O	TYPE	DIM	DEFINITION
INPUT	I	I	VARB	BUFFER CONTAINING THE DIRECTIVE GIVEN FOR THE SEMI,AUTO OR AUTO* MODE. FLAG SHOWING PRESENCE OF A SYNTAX ERROR IF IT IS NONZERO.
STATUS	I	O	I	

EXTERNAL REFERENCES
NONE

DIAGNOSTICS
*** SYNTAX ERROR
THE USER MADE A SYNTAX ERROR WHEN HE ENTERED THE DIRECTIVE.

EXTERNAL STORAGE
NONE

BLANK COMMON
VARB I/O

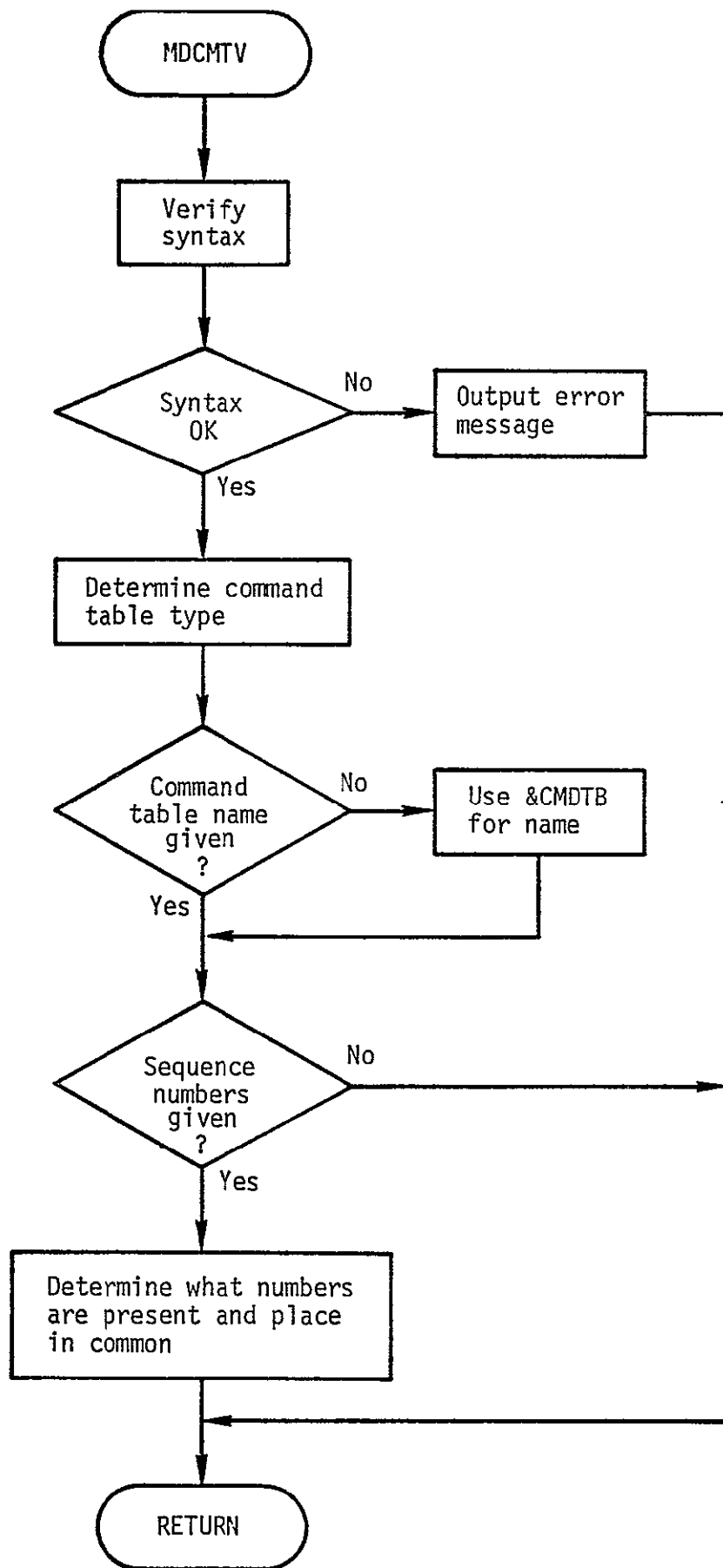
CMDNO 0
CMTNAM 0
CMTYP 0
DIRECT I
ENDNO 0

COMMON / MDCODE /
VARB I/O

ASTRIC I
COMMA I
DOLLAR I
EOS I
INTGR I
MINUS I
NAME I
PERCNT I
UPARRW I

LOCAL COMMON

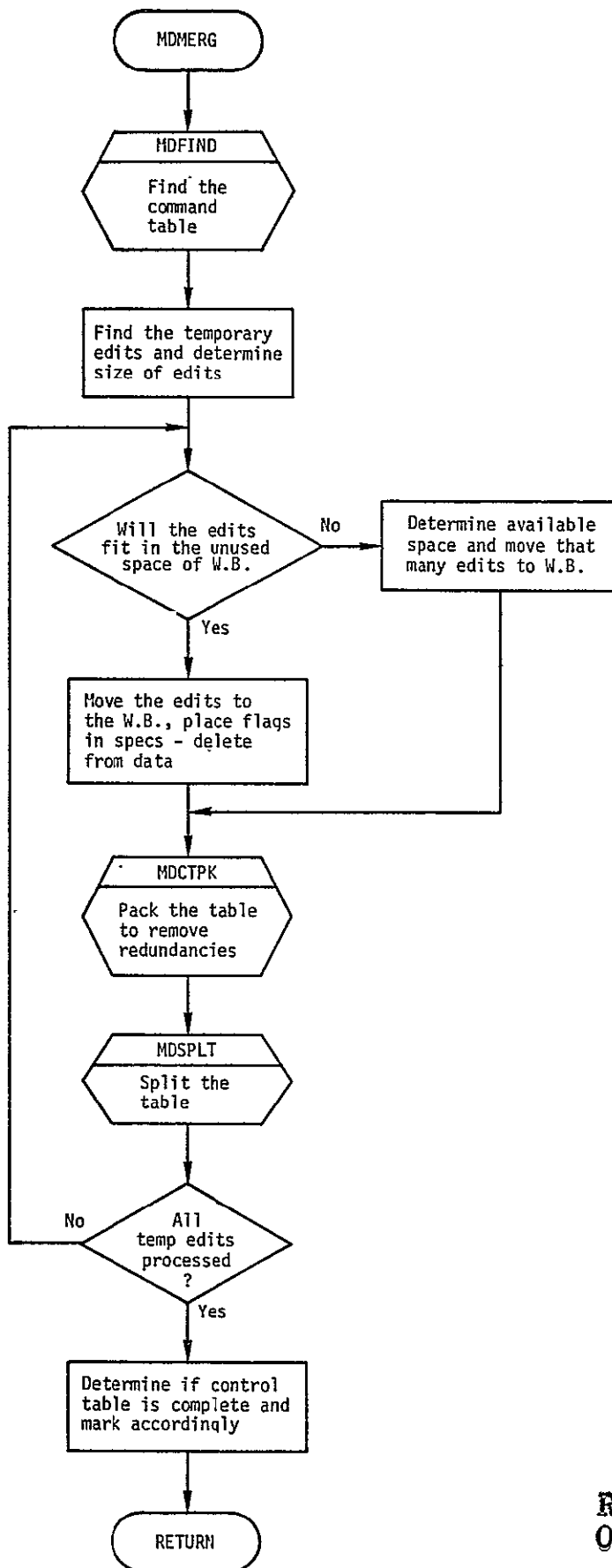
NONE



MDCMTV Flow Diagram

MDMERG - Execution Controller

MDMERG, when developed and implemented, will locate any applicable temporary edits within the command table being executed and modify the control table accordingly before the particular command is executed. The accompanying flowchart is a functional representation of MDMERG's task.



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDPRT - Execution Controller

MDPRT, when developed and implemented, will print the control table variables at execution time which have been designated to be printed. The ** feature of the control level syntax is used to designate execution time printing of a control table variable. Input variables are printed before processor execution and output variables are printed after processor execution.

MDSMON - Submonitor/Execution Controller

MDSMON is the main subroutine to the submonitor/execution controller component. This routine is used to assist the user in constructing and executing a simulation. MDSMON's execution controller function is to receive user inputs regarding the commands to be executed, call the control table editor to complete the control table if necessary, and call MDALOC to establish the proper linkages for all arguments input to and output from each processor executed.

Method

Input: The inputs to MDSMON are obtained from the user. The user is prompted for directives and in the MANUAL mode commands are input, in the SEMI mode the user is prompted to verify or to change the command. In the AUTO mode there is no user interaction unless an incomplete control table is encountered.

Processing: MDSMON is entered from the MDAS resident to control the execution of a simulation. On the initial entry into MDSMON the user is prompted for an access code and whether or not to initialize the data area. The user is then prompted with an up arrow for a directive. Valid directives are:

Utility directives	{	USAGE	Gives elapsed time of MDAS session
		COPY	Not operational
		QUIT	Terminates session
		EDCMT	Edits command table
		EDCNT	Edits control table
		TOC	Lists table of contents of the information elements in user data area
		DELETE	Deletes data area in SMT
		DUMP	Lists contents of an information element
		SAVE	Save data area (SMT)
		RECALL	Recall saved data area (SMT)
Execution directives	{	SEMI	Executes commands in command table but allows user to alter commands before execution
		AUTO	Commands in command table are executed without any user interaction
		MANUAL	Execution controller prompts user for each command
		AUTO*	Same as AUTO but commands are printed as they are executed.

For each utility directive, MDSMON calls a subroutine to perform the designated function and then returns for another directive (except in the case of QUIT).

The execution directives are processed within MDSMON. When one of these modes is entered, MDSMON processes a command and returns to the resident for execution. When MDSMON is reentered, the mode remains in effect until all commands are processed and executed at which time the user is prompted for a new directive.

In the manual execution mode, MDSMON prompts the user for each command to be executed. The end of the execution sequence is determined when the user responds with an "+" or fails to enter a new command (i.e., presses carriage return in response to the command prompt "#"). In the manual mode MDSMON will verify and interpret the syntax of each command entered.

In the automatic and semi-automatic execution modes the name of a command table is input and MDSMON will retrieve this information element from the storage monitor table (SMT). Once the command table has been stored in the working command table (&CMDTB), its commands are processed sequentially. Each command to be executed is extracted from &CMDTB using an index stored in non-volatile memory. An optional field of the SEMI and AUTO directives allows the user to specify the range of commands or the beginning command in the execution sequence. The sequence number input in that field determines the initial value of the command table index.

In the automatic execution mode the processors specified in each of the commands are executed with no user interaction unless an incomplete control table is encountered. An option on the AUTO directive has MDSMON indicate its progress by printing each command as it is executed.

In the semi-automatic execution mode the controller prompts the user with the sequence number, processor name and control table name of each command. In response to this prompt, the user has five options:

1. Carriage return, giving concurrence to execute the command.
2. "# nnn", directing the execution controller to a different command in the table (nnn is its sequence number).

3. "#", indicating that a manual override command is to be input in place of the prompted command.
4. "\", indicating that temporary edits are to be made to the control table before executing this command.
5. "+", the SEMI mode is to be aborted and control returned to the directive level.

In each of the execution modes, MDSMON checks the control table specified in each command for completeness and for consistency with the current version of the processor to be executed. A revision number is retained in the processor catalog (PROTAB) for each processor and updated only when the processor interface changes. This revision number is also placed into each control table when it is created.

If an incomplete control table is found MDSMON calls the control table editor (entry MDEDCN) for the purpose of interacting with the user to complete this table. In the MANUAL and SEMI modes a syntax mechanism ("\ following the command) exists for directing the execution controller to call MDEDCN even if the control table is complete.

For each processor to be executed MDSMON must also establish the input and output arguments' linkages. This is accomplished by calling MDALOC which also sets up the parametric scan control data. If the processor is a utility, MDALOC will not be entered. The utility processor will be called instead and will set up its own input and output argument linkage. Currently only one utility processor exists, MDALCT, which performs the ALOCAT command (allocates an array).

Output: The output from MDSMON is dependent on the input directive. If the directive is other than SEMI, AUTO or MANUAL, the designated function is performed. If MDSMON is in the SEMI, AUTO or MANUAL mode, the control table is edited if it is incomplete or is specified on input for edition. The argument linkage is established for the processor before returning to the resident for execution of the command.

SAGE

ENTRY MDSMON
CALL MDSMON

EXTERNAL REFERENCES

MDLOGO
MDPRMT
MDSMTR
MDCMT
MDCNT
MDUTIL
MDQUIT
MDVCMO
SEARCH
MDGETC
MDGET
MDSPLT
MDMERG
MDFIND
MDCMTL
MDTOC
MDCMTV
MDCMTG
MDCMTS
MDSMTW
OBEY
MDEDCN
MDALOC
MDCONV
MDALCT

DIAGNOSTICS

UNDEFINED DIRECTIVE

RESPONSE TO UP ARROW WAS NOT LEGAL.
IT MUST BE A DEFINED DIRECTIVE

PROCESSOR NAME NOT FOUND

PROCESSOR NAME NOT FOUND IN THE PROCESSOR CATALOG

CONTROL TABLE NAME NOT FOUND

CONTROL TABLE NAME NOT FOUND IN THE CONTROL TABLE

*** REVISION NO. OF(..) DOES NOT MATCH

REVISION NO. OF CONTROL TABLE (..)

THE REVISION NUMBER IN THE CONTROL TABLE DID NOT
MATCH THE REVISION NUMBER FOR THIS PROCESSOR IN THE
PROCESSOR CATALOG

***ILLEGAL RESPONSE

IN THE SEMI-AUTOMATIC MODE, AN ERROR OCCURED IN THE
PROMPTING OR THE USER'S FIRST INPUT WAS NOT A POUND,
BACKLASH OR A CARRIAGE RETURN.

EXTENT OF (.....) IS TOO LARGE FOR

CURRENT MDAS CONFIGURATION (.....).

CURRENT PROCESSOR WILL NOT FIT INTO THE REMAINING
PORTION OF MEMORY

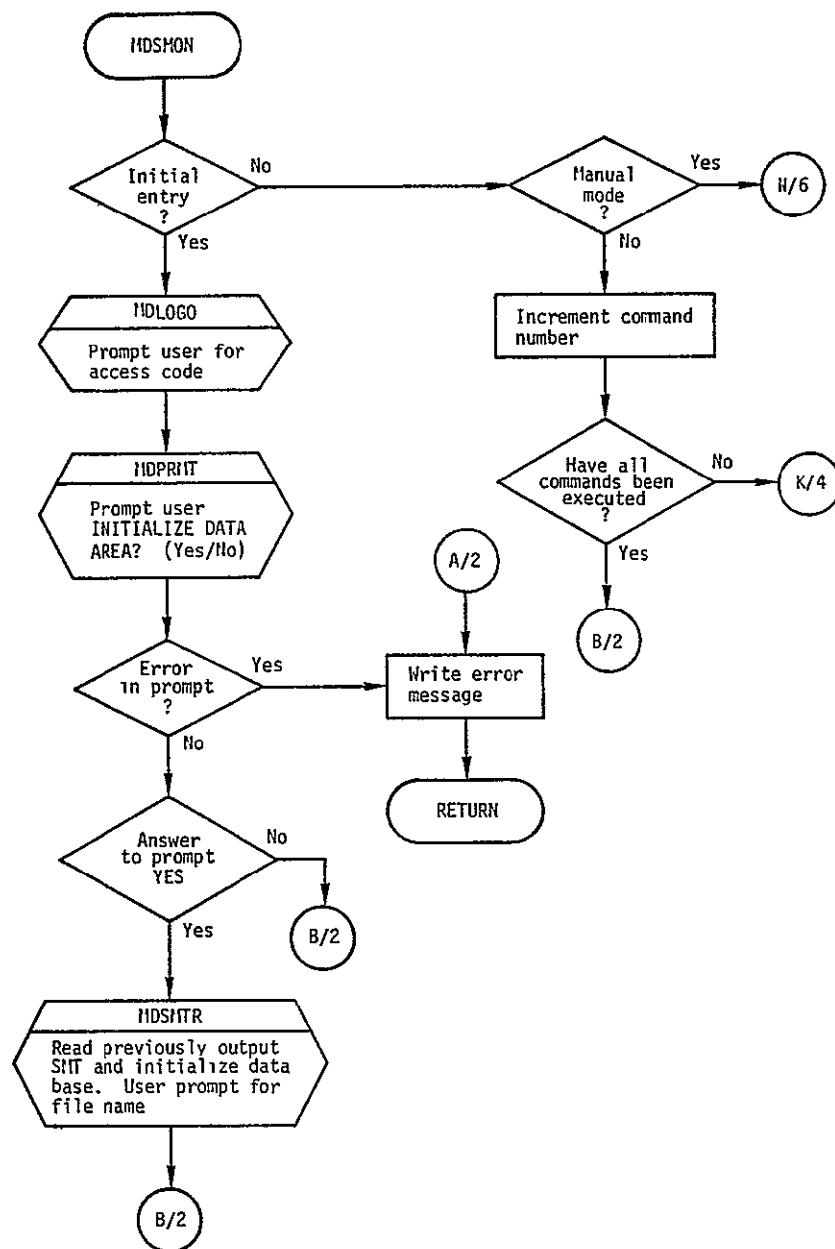
*** ERROR IN ALOCAT -- STATUS= ...

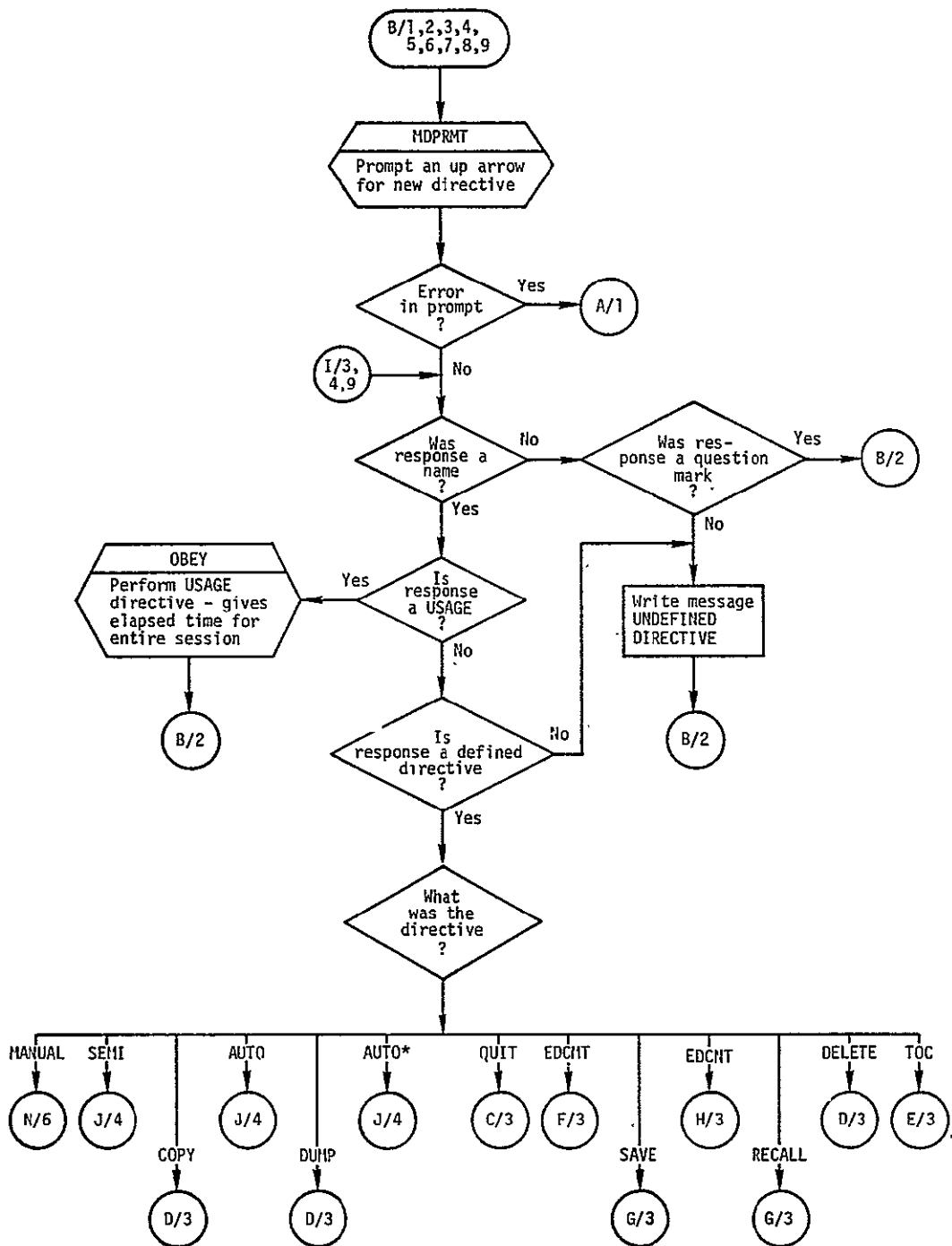
ERROR OCCURED IN ROUTINE MDALCT

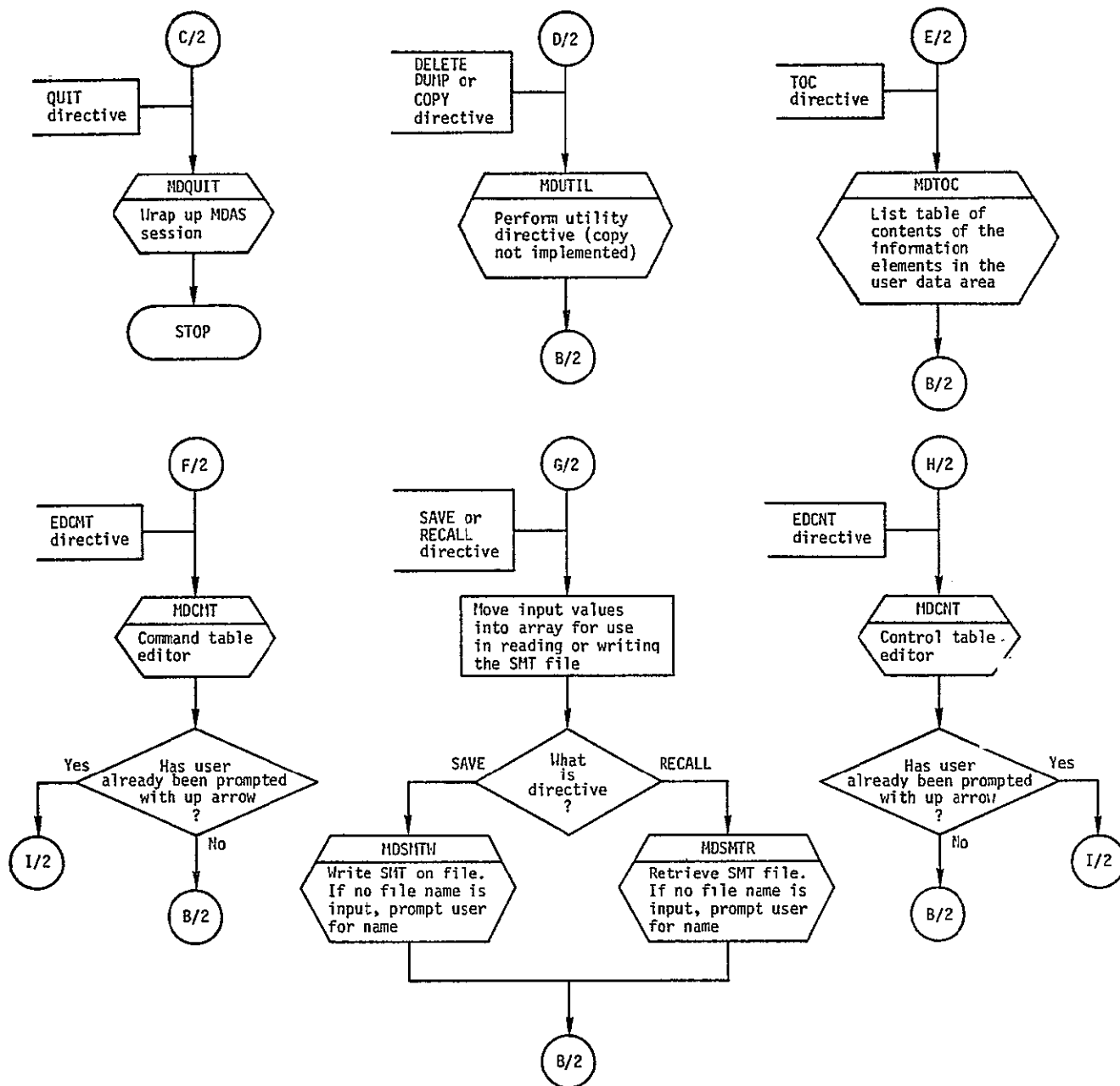
I/O ERROR WHILE PROMPTING

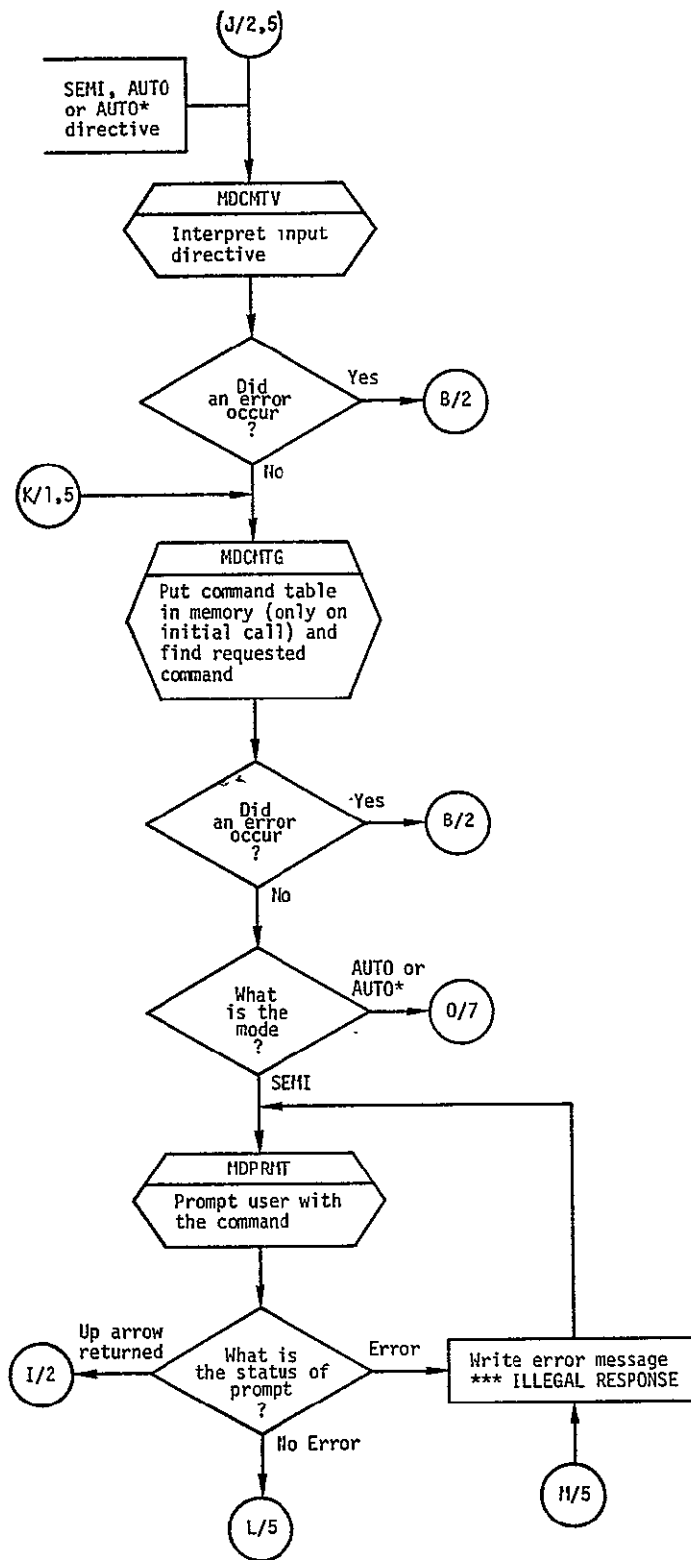
ERROR OCCURED IN ROUTINE MDPRMT

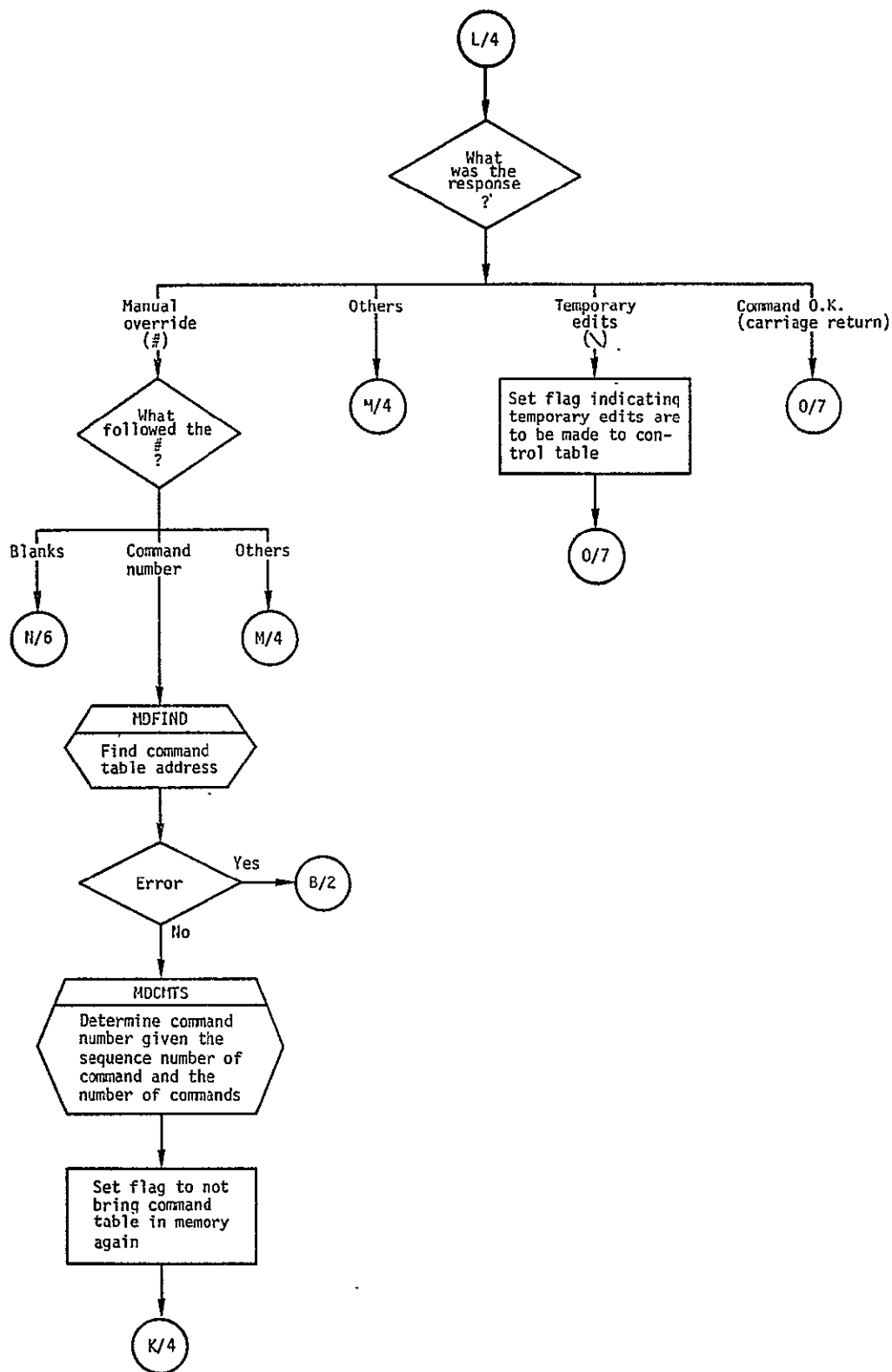
EXTERNAL STORAGE	
NONE	
BLANK COMMON	
VARB	I/O
CMDNO	I/O
CMTNAM	I
CMTYP	I
CTNAME	O
CTYPE	O
DIRECT	O
DBANK	I
EDIT	I
ENDNO	I
ENTRY	I/O
IBANK	I
PNAME	O
PRINT	O
PRONAM	O
PRONUM	O
PROTAB	I
PTABKY	I
SEQNO	O
COMMON /MDBUFF/	
VARB	I/O
BDATA	O
DSIZE	O
MDLEN	O
WBUF	O
COMMON /MDCODE/	
VARB	I/O
ASTRK	I
BKSLSH	I
EOL	I
INTGR	I
NAME	I
POUND	I
QSTION	I
LOCAL COMMON	
NONE	

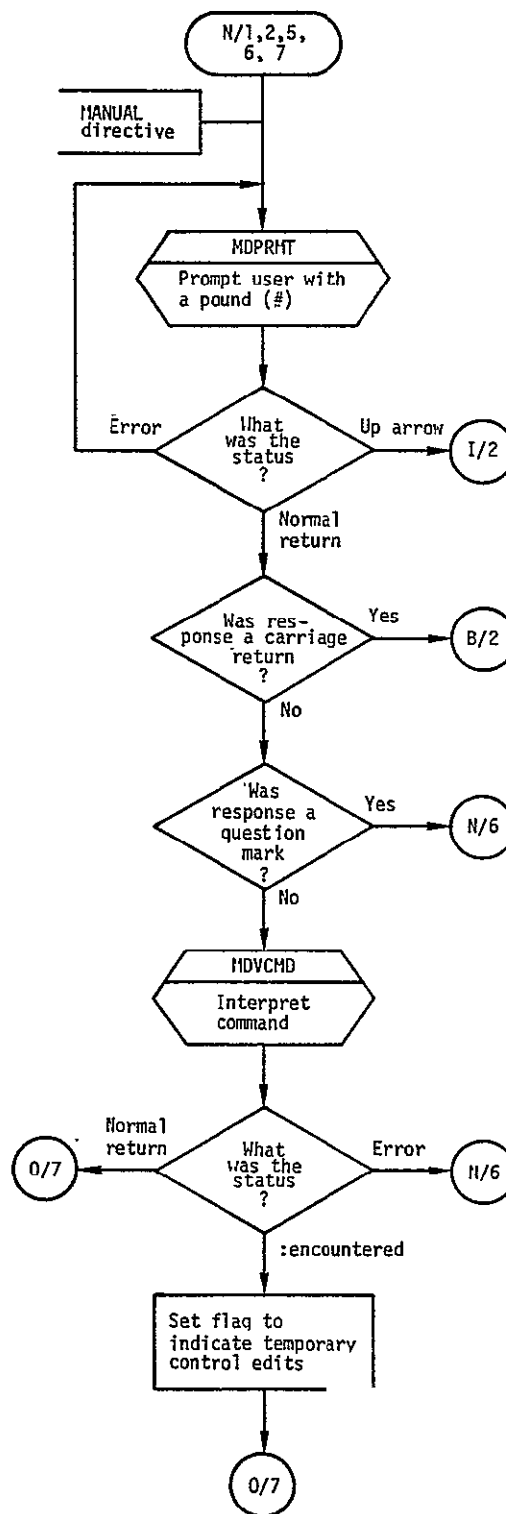


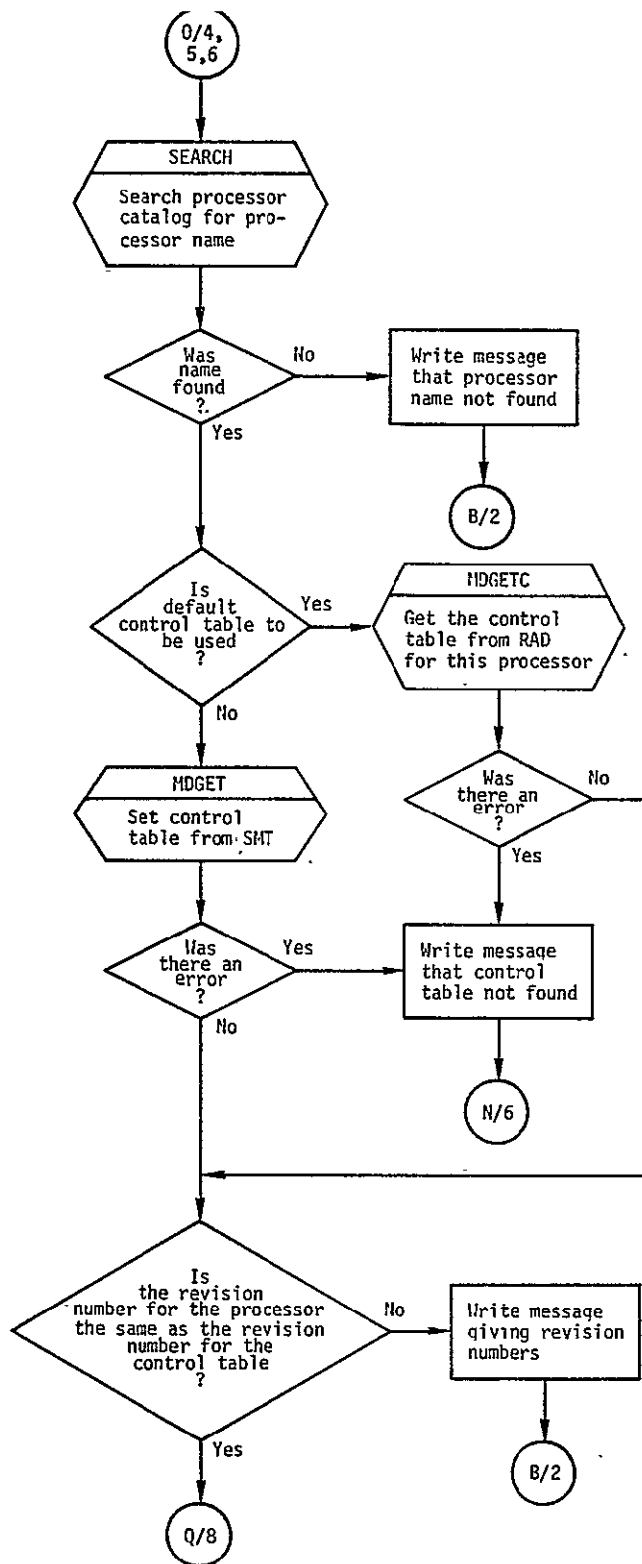


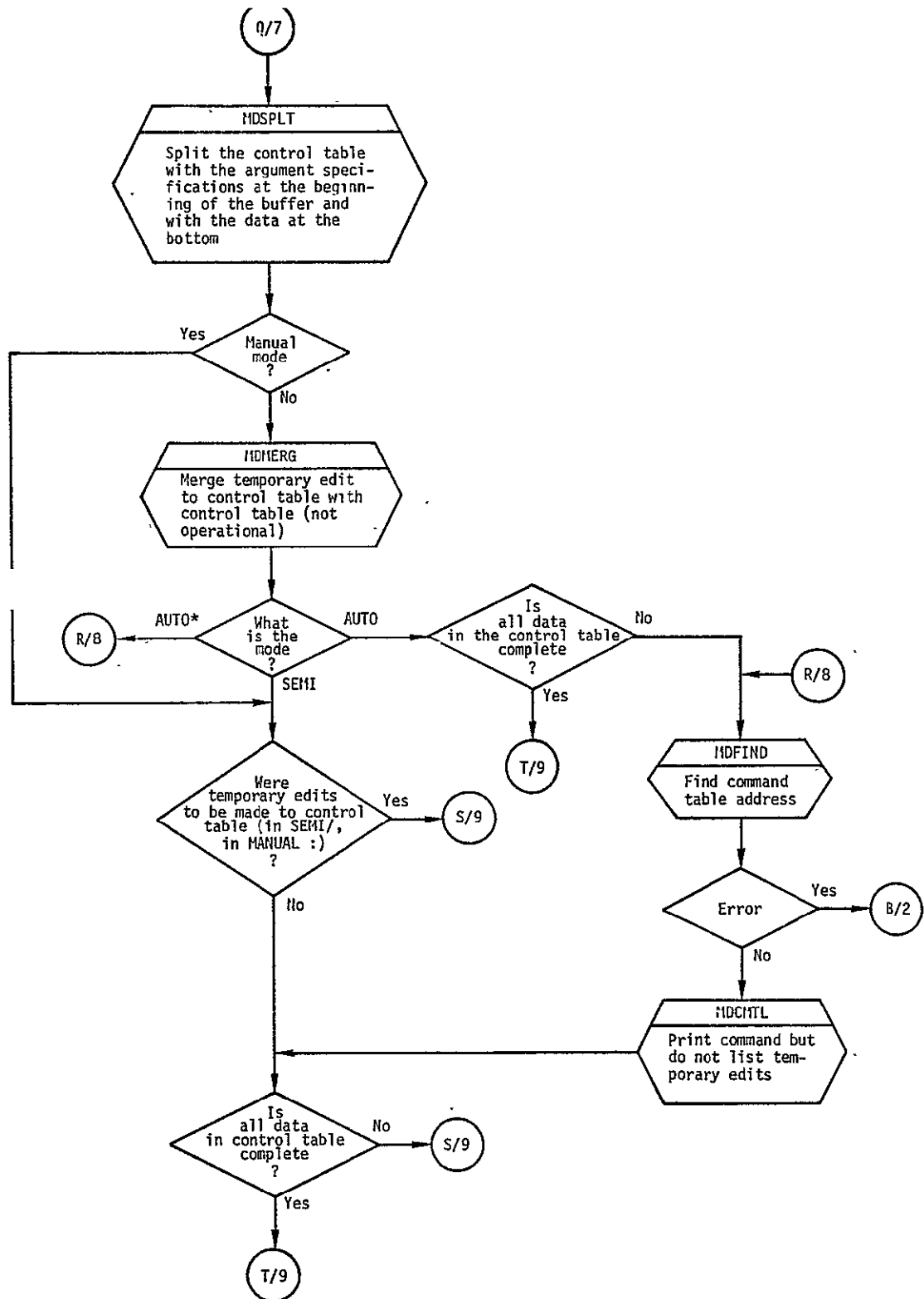


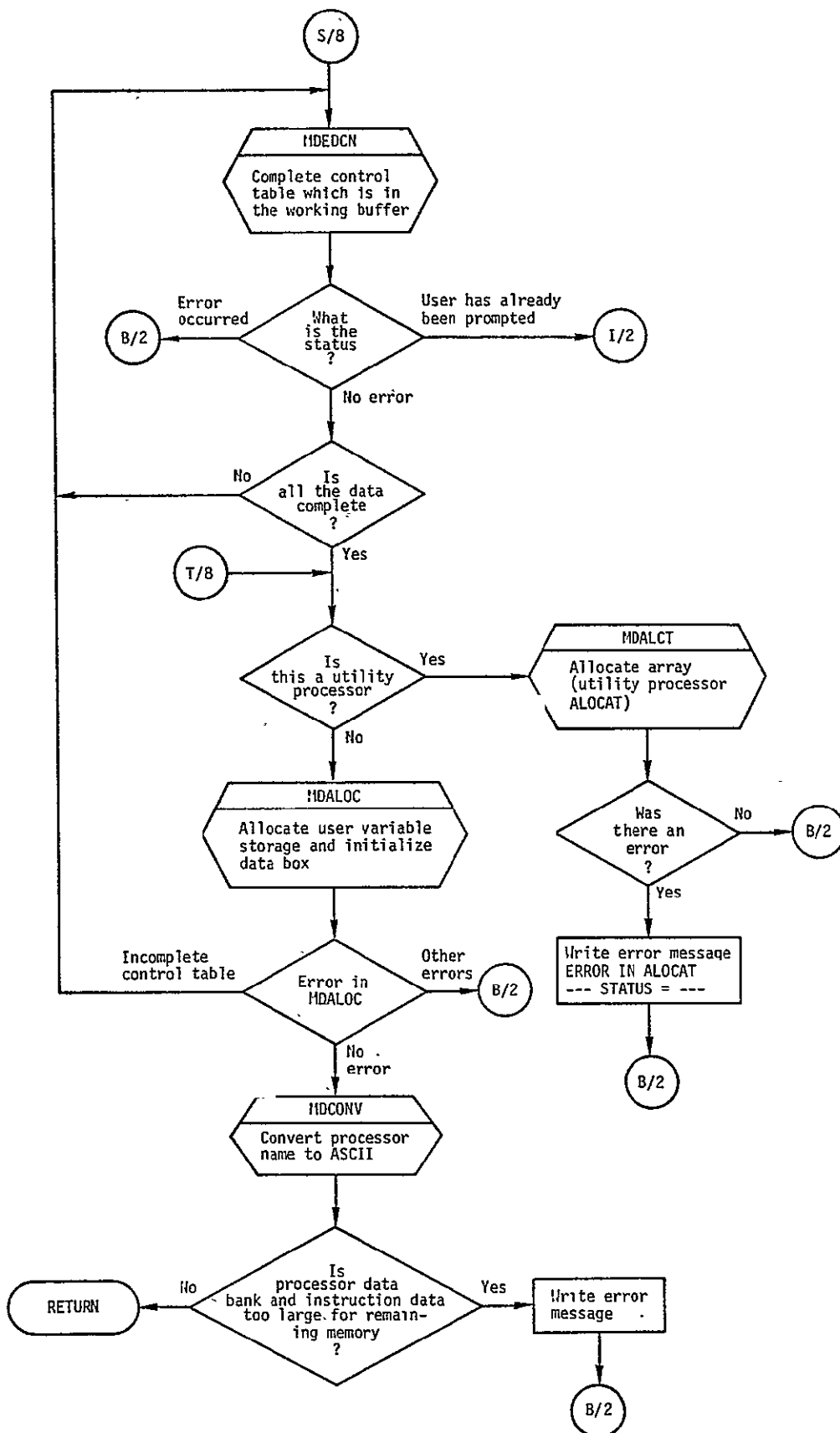












MDCMNT - Command Table Editor

MDCMNT is the interfacing routine between the command table editor (MDCMT) and the control table editor (MDCNT) for the appending of temporary control table edits to a command. It maintains the order of the working buffer, modifies the data from what MDCNT provides and deletes specified edits when required.

Method

Input: Upon entry, the command table is contained in the working buffer and is split into two parts: the commands and any existing temporary edits. In addition, the processor name and the sequence number of the command to be edited as well as an indicator showing the existence, or lack, of previous temporary edits for this command are input through the calling sequence.

Processing: A test is made to determine if edits already exist for this command. If they do, they may be either retained or deleted. If deletion is requested, the edits are removed and the remaining edits packed. Control is returned to MDCMT with a successful edit status. If the edits are to be retained, they are moved to the top of the temporary edits, their heading removed, packed and processing is continued as though they were performed at this time. If no edits existed, or we are retaining them, a search is made to determine if the processor exists. If it does not, control is returned to MDCMT immediately after posting an error message. Otherwise, the revision number of the processor and the length of the default control table are obtained.

In order to utilize the control table editor the argument specifications must be brought into the working buffer. Therefore, a portion of the command table in the buffer is written to the SMT with the name &CMDTB and the specifications read in to the buffer in their place. All arguments are then marked undefined and control is given to the control table editor (entry MDCNTM).

Upon return from the control table editor, the edits just made are packed by deleting any duplication in arguments. In addition, specified flags and data type are placed in the argument label field of the data and the argument

name is placed with the data. A heading is placed "on top" of the edits which consists of the sequence number of this command, the revision number, number of edits and the length (in words) of the edits.

That portion of the command table which was written to the SMT upon entry is retrieved and placed back in the working buffer. The entry in the SMT for &CMDTB is deleted and control is returned to MDCMT.

Output: The only outputs from MDCMNT are a status flag indicating how successful the edits were and any new edits made. The working buffer remains split. If the user has entered "up arrow" (↑) while under MDCMNT control, the directive entered is contained in a prompting buffer in the calling sequence and the status flag is set to so indicate.

USAGE

ENTRY MDCMNT

CALL MDCMNT (PNAME,SEQNO,FLAG,INPUT,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
PNAME	I	I	1	THIS IS THE PROCESSOR NAME FOR WHICH TEMPORARY EDITS ARE TO BE MADE.
SEQNO	I	I	1	THIS IS THE SEQUENCE NUMBER OF THE COMMAND TO WHICH TEMPORARY EDITS ARE TO BE MADE.
FLAG	I	I	1	THIS IS AN INDICATOR WHICH DETERMINES THE PRESENCE OF PREVIOUS TEMPORARY EDITS: =0,NO PREVIOUS EDITS =1,PREVIOUS EDITS DO EXIST BUT RETAIN THEM =2,PREVIOUS EDITS EXIST BUT DELETE THEM.
INPUT	O	I	50	THIS BUFFER WILL CONTAIN THE DIRECTIVE ENTERED BY THE USER SHOULD HE TERMINATE EDITING BY ENTERING AN UPARROW
STAT	O	I	1	THIS IS AN INDICATION OF WHAT OCCURRED IN MDCMNT: =1,ERROR OCCURRED IN READING OR WRITING A FILE OR PNAME DOES NOT EXIST =0,TEMPORARY EDITS PERFORMED SUCCESSFULLY =-1,CONTROL TABLE EDITOR (MDCNTM) ENCOUNTERED AN ERROR WHILE DOING EDITS =2,WORKING BUFFER OVERFLOW,OR USER ENTERED UPARROW (↑)

EXTERNAL REFERENCES

MDCNTM
MDELET
MDGET
MDGETC
MDPUT
SEARCH

DIAGNOSTICS

- EDCNT ERROR
THE CONTROL TABLE EDITOR HAS ENCOUNTERED AN ERROR WHILE DOING THE EDITS (STAT=-1)
- PROCESSOR NAME NOT FOUND
THE PROCESSOR NAME SPECIFIED COULD NOT BE FOUND IN PROTAB (STAT=1)
- UNABLE TO READ DEFAULT CONTROL TABLE
WHEN ATTEMPTING TO READ THE ARGUMENT SPECS FROM THIS PROCESSORS DEFAULT CONTROL TABLE A FATAL ERROR OCCURRED (STAT=1)
- UNABLE TO READ SMT
ATTEMPTING TO READ THE COMMAND TABLE FROM THE SMT RESULTED IN A FATAL ERROR (STAT=1)
- UNABLE TO WRITE TO SMT

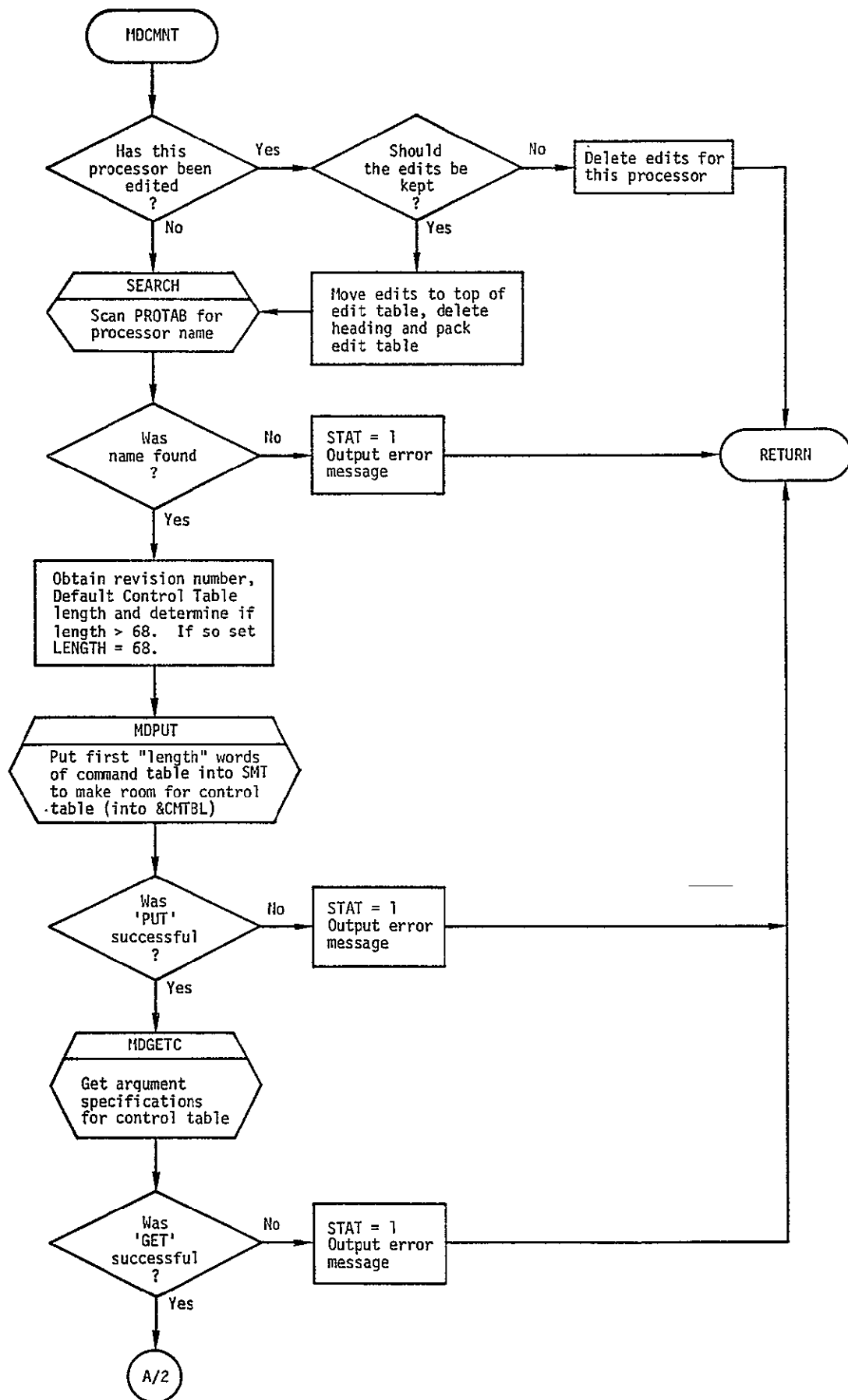
AN ERROR OCCURED WHEN WRITING THE COMMAND TABLE TO
 THE SMT (STAT=1)
 ●●WORKING BUFFER OVERFLOW-EDITING ABORTED
 THE USER HAS ATTEMPTED TO MAKE TOO MANY EDITS AND
 THERE IS NOT ENOUGH ROOM IN THE WORKING BUFFER FOR
 THEM (STAT=2)
 ●●--NO TEMPORARY EDITS PERFORMED--●●
 THIS MESSAGE OCCURS WHEN ANY OF THE ABOVE ERRORS
 OCCUR.

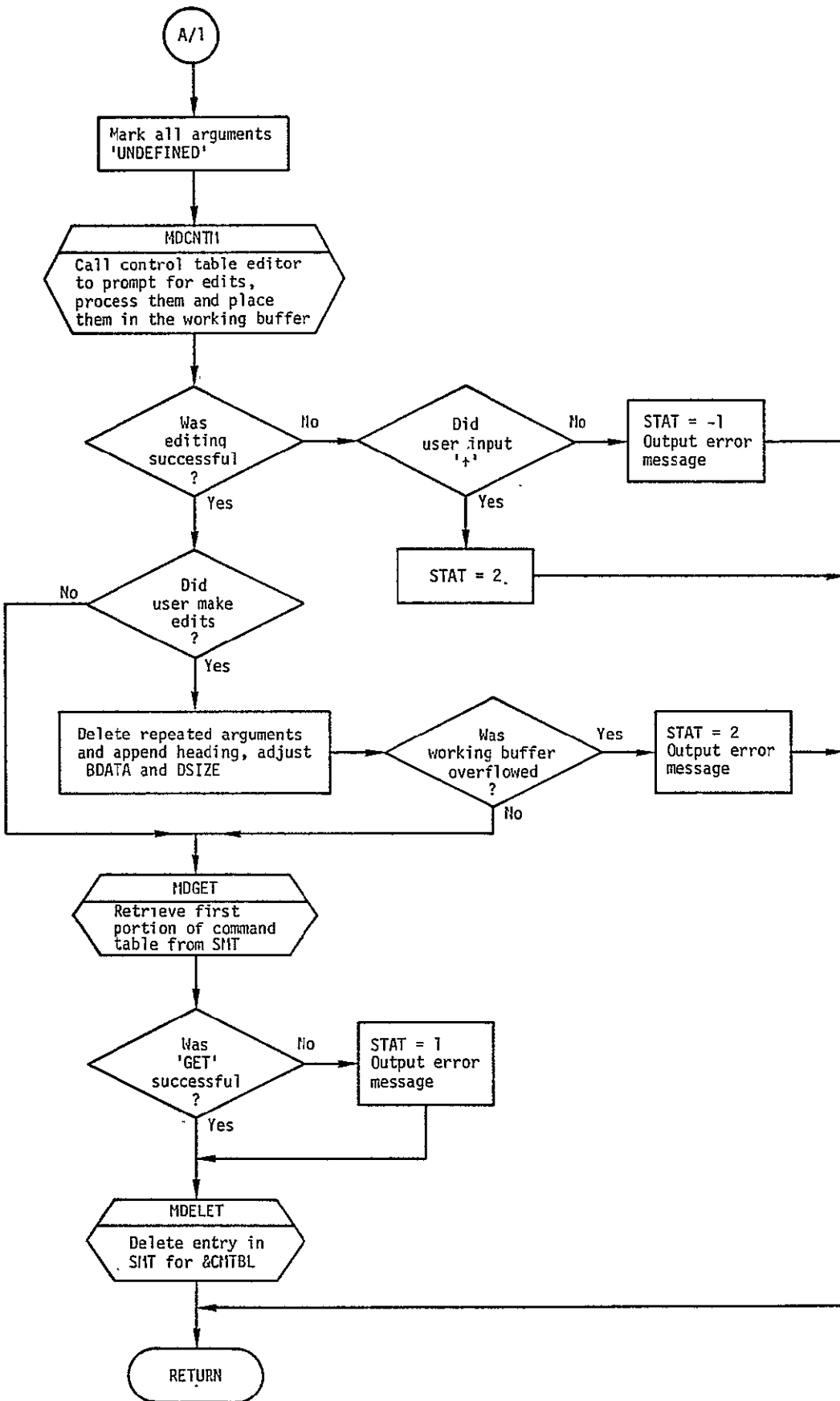
EXTERNAL STORAGE
 NONE

BLANK COMMON
 VARB I/O

PBLN I
 PROTAB I

COMMON / MDBUFF /
 BDATA I/O
 DSIZE 0
 WB I/O





MDCMT - Command Table Editor

MDCMT is the driving routine of the command table editor. Its primary function is to interact with the user at the command level to allow him to build, or modify, command tables or to append temporary edits to the control table of an existing command.

Method

Input: The EDCMT directive, with its optional fields, is the only input to MDCMT. The submonitor calls MDCMT with this directive in the prompting buffer after it has been processed by MDSCAN.

Processing: MDCMT interprets the input directive and determines what fields are present. If needed, it obtains a requested command table from the SMT and, in any event, begins prompting the user to determine what options are to be performed. The user may perform any of four options: list, number, delete or enter a command.

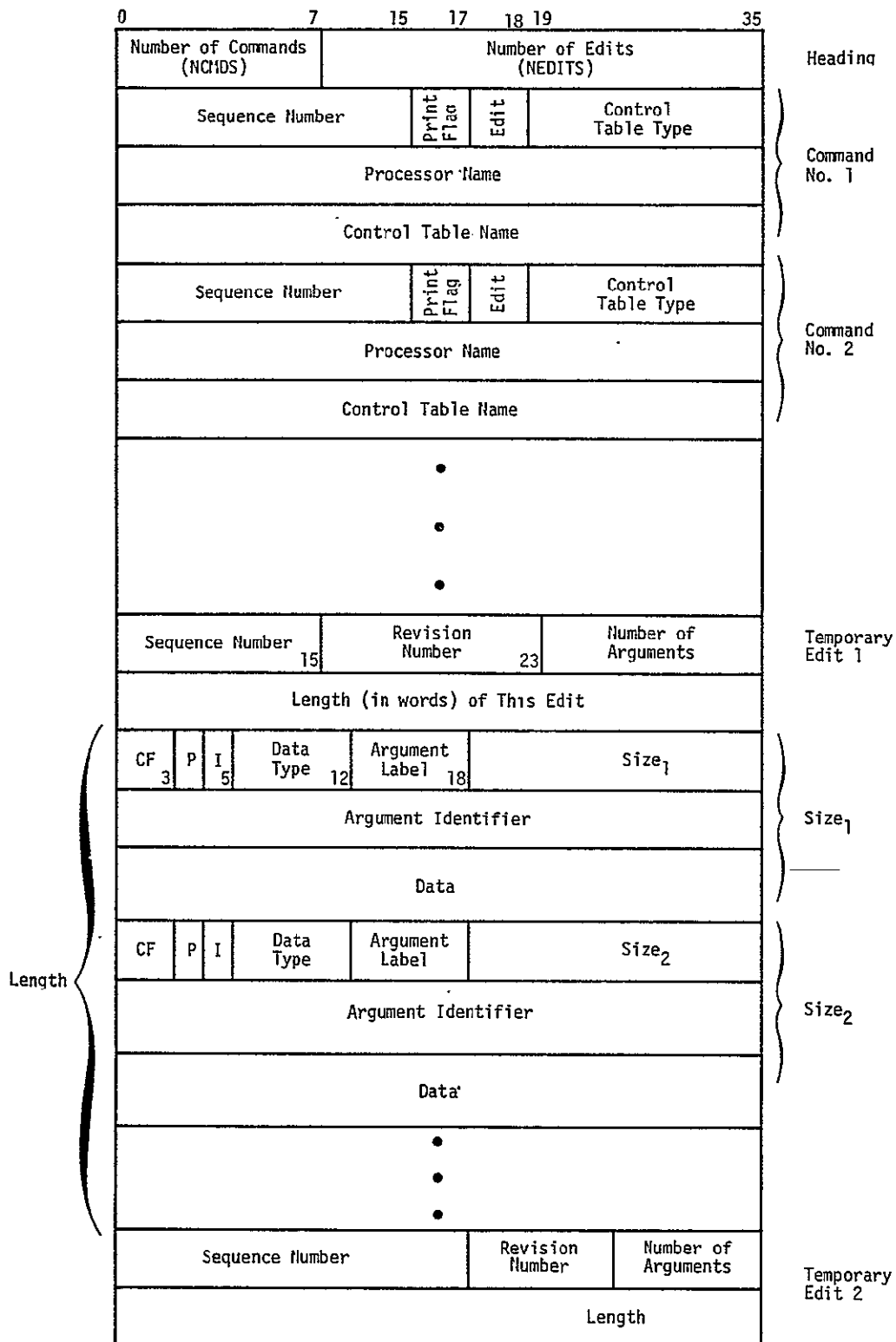
The list option allows the user to list all, or a specified portion, of a command table. The user performs this by entering LIST START,END where START is the beginning sequence number to start listing from and END is the last sequence number to be listed. If neither START nor END is input the entire command table is listed. MDCMT determines if both START and END are existing commands and, if not, informs the user of such. If the commands can be listed, MDCMT calls MDCMTL to perform the list. If any of the listed commands have temporary control table edits appended to them, the edits are listed immediately following the command by MDCMTL. After completing this option MDCMT again prompts the user to allow him to perform another option.

The number option is performed by the user simply entering NUMBER. At this time MDCMT renumbers all commands in the table. The first command is given sequence number 100 and each successive command is 100 more than the command preceding it. Any temporary control table edits that exist are also renumbered to retain the correlation between the edit and the command. After completion the user is again prompted to allow him to request another service.

The user performs the delete function by entering -START,END where START is the sequence number from which to begin deleting and END is the sequence number of the last command to delete. If a one or both of the sequence numbers do not exist, the user is notified of such and reprompted. If the function can be performed, all the commands between, and including, START and END are deleted as are any temporary edits which existed for a deleted command. Once again, the user is prompted, upon completion, for another option.

The command option allows the user to add new, or modify existing, commands. MDCMT automatically prompts with sequence numbers anytime the user is building a new table or begins inserting commands in the table past the last one currently in the table. In other cases the user is prompted only with #. The prompted sequence number will be modulo 100 and will be 100 greater than the last automatically prompted command. The user terminates the automatic prompting by depression of carriage return without entering a command. At this time the user is prompted with # to allow him to insert new commands, modify existing ones or perform any of the options described above. If the user modifies a command which has temporary edits appended, the edits are deleted. The user specifies temporary edits by appending a colon (:) to the control table name present in the command. Upon completion of editing, the command table is sorted, packed and placed into the SMT and assigned the name provided by the user, or &CMDTB if no name was provided.

Output: If the user terminates the command table editing normally (i.e., depresses carriage return after being prompted with #) a status flag (indicating normal completion) is the only output. However, if the user terminates by entering "up arrow" (↑) and enters a directive, not only is a status flag indicating this fact output, but also the prompting buffer containing the directive is output to the submonitor (MDSMON).



Command Table Format (Packed)

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

Command Table Definitions

Commands: Total of $3 * \text{NCMDS} + 1$ words

Print Flag - A flag indicating the level of Control Table print at execution.

Edit - Indicates the presence of temporary edits for this command.

Temporary Edits

CF - Completion flag;
= 0 Argument is currently undefined
= 1 Incomplete
= 2 Complete

P - Print flag;
= 0 Argument not to be printed at execution time
= 1 Printed

I - Indicator;
= 0 Immediate data (=)
= 1 Execution time data (@)

USAGE

ENTRY MDCMT
CALL MDCMT (INPUT,STATUS)

ARGMT	I/O	TYPE	DIM	DEFINITION
INPUT	I/O	1	50	UPON ENTRY THIS BUFFER CONTAINS THE COMMAND(SCANNED) !EDCMT (OLDNAME), (NEWNAME). IT IS USED INTERNALLY AS A PROMPTING BUFFER AND IS AN OUTPUT ONLY WHEN THE USER ENTERS A NEW DIRECTIVE WHILE MDCMT IS PROMPTING
STATUS	O	1	1	THIS VARIABLE IS ONLY USED WHEN THE USER INPUTS A NEW DIRECTIVE WHILE MDCMT IS IN CONTROL. IT TELLS MDSMON A DIRECTIVE HAS BEEN ENTERED.

EXTERNAL REFERENCES

MDCMNT
MDCMTL
MDCMTS
MDGET
MDPRMT
MDPUT
MDSPLT
MDVCMD
SORTI

DIAGNOSTICS

***COMMAND TABLE COULD NOT BE SAVED
THE NAMED COMMAND TABLE COULD NOT BE INSERTED
IN THE SMT.

***** IS NOT AN EXISTING COMMAND TABLE
THE USER HAS ENTERED THE NAME OF A COMMAND TABLE TO
BE MODIFIED WHICH COULD NOT BE FOUND IN THE SMT.

***SYNTAX ERROR
THE USER HAS ENTERED INCORRECT SYNTAX FOR ONE OF
THE OPTIONS AVAILABLE UNDER THE EDCMT DIRECTIVE.

***UNDEFINED SEQUENCE NUMBER
THE USER HAS SELECTED THE LIST OPTION BUT ONE,OR
BOTH, OF THE SEQUENCE NUMBERS HE HAS INPUT DOES NOT
EXIST.

***WORKING BUFFER OVERFLOW--EDITING ABORTED. COMMAND TABLE
***** SAVED BUT MAY NOT BE COMPLETE
THE USERS MODIFICATIONS REQUIRED THAT MORE SPACE THAN
IS AVAILABLE BE USED. THE TABLE IS SAVED BUT SHOULD
BE USED ONLY AFTER BEING COMPLETELY CHECKED.

EXTERNAL STORAGE
NONE

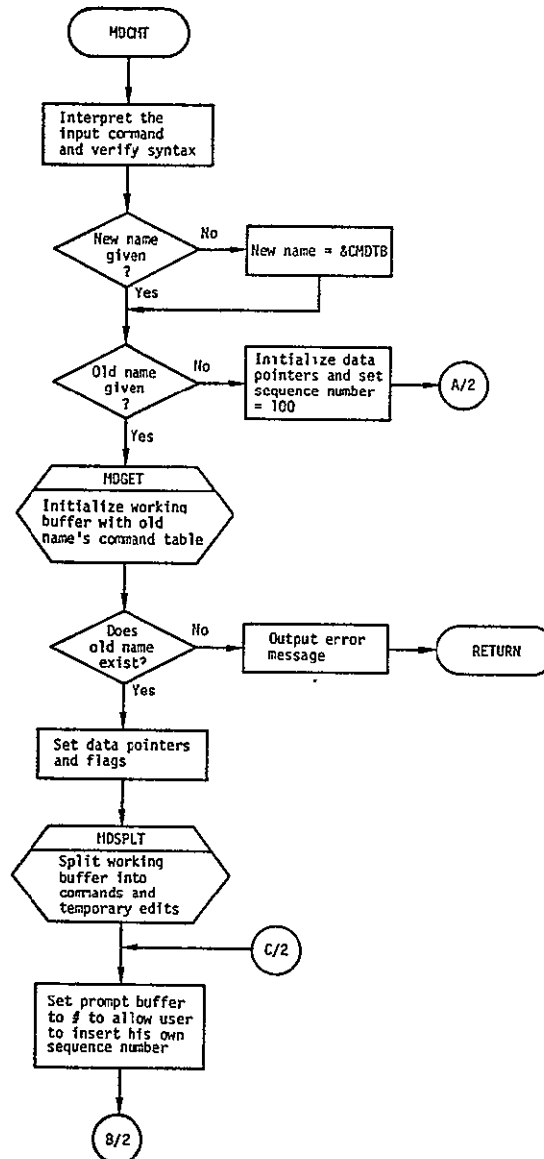
COMMON / MDBUFF /
MDLEN . 1
BDATA I/O
DSIZE I/O
WBUF

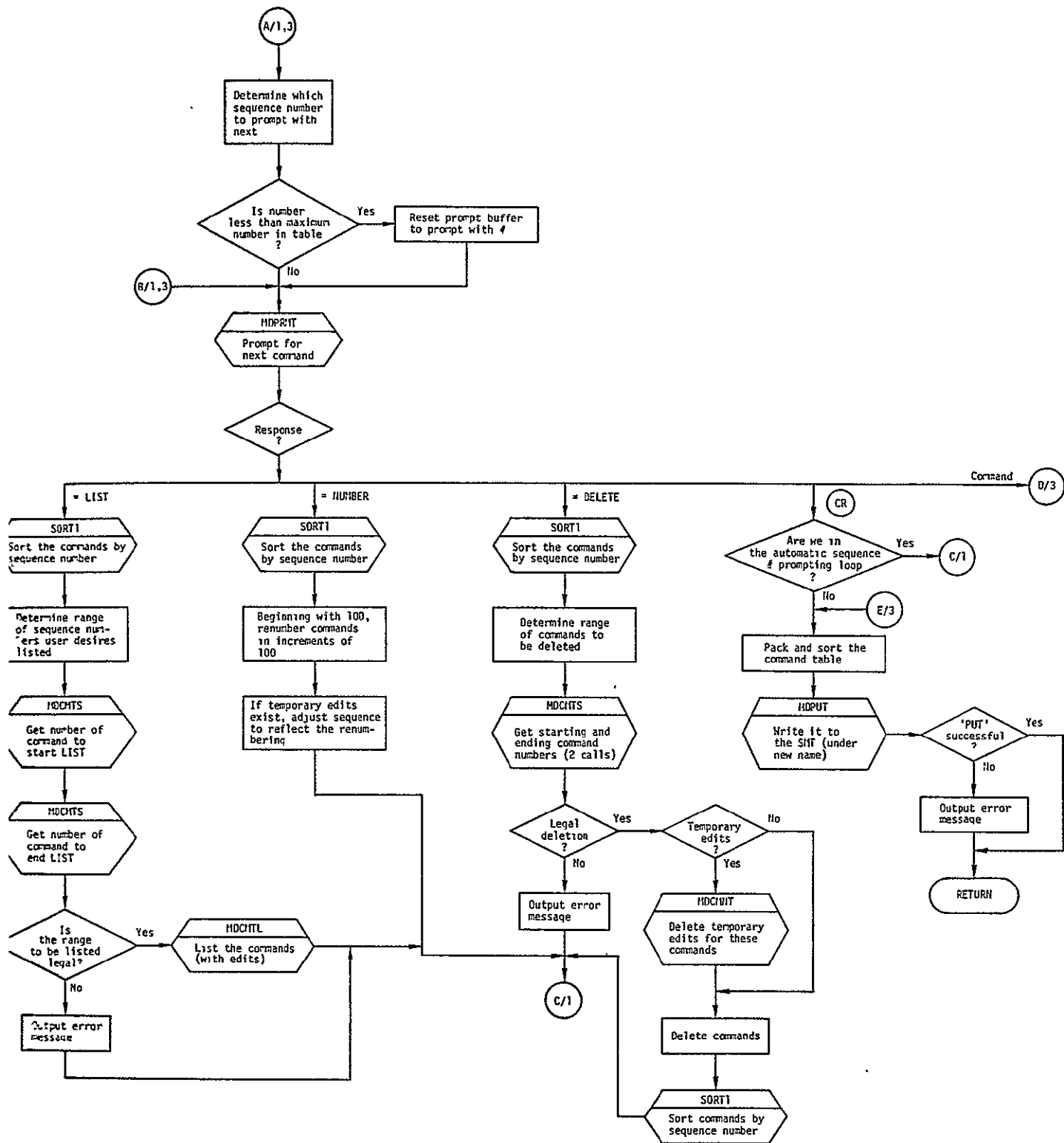
COMMON / MDCODE /
NAME 1
INTGR 1

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

EOS	I
DOLLAR	I
PERCNT	I
COMMA	I
UPARRW	I
MINUS	I
POUND	I

BLANK COMMON
NONE

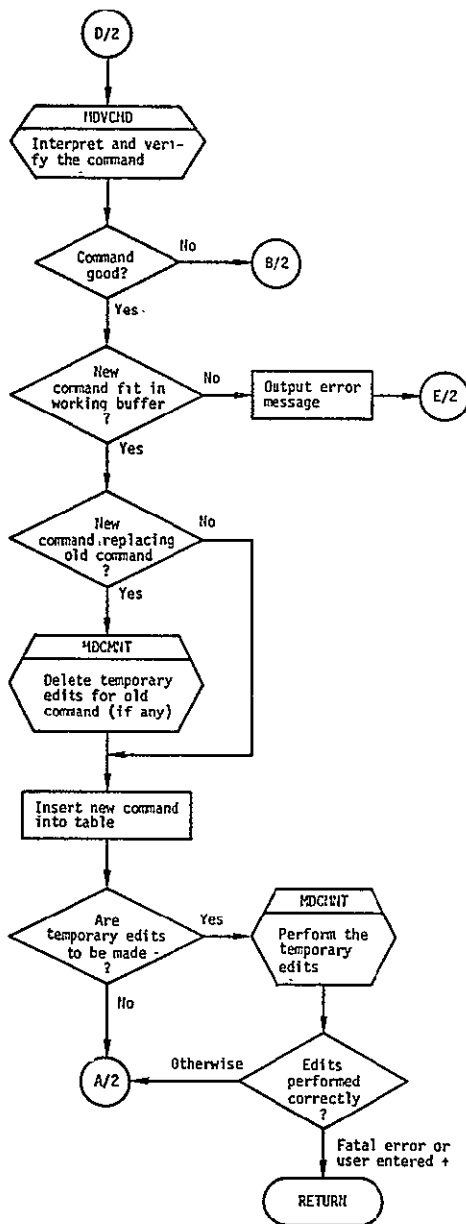




MDCMT Flow Diagram Page 2 of 3

6.2-8

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDCMTL - Command Table Editor

MDCMTL is the routine which is used to print the command table. It is structured such that all or part of the table may be printed.

Method

Input: All input is contained in the calling sequence and consists of: the number of the command at which the print begins, the number at which to stop the print, a flag indicating if called by the DUMP directive, if called by the command table editor or if the temporary edits are not to be listed, and a buffer containing the command table.

Processing: Each command to be listed is broken down into its constituent parts and printed. As each command is printed, a check is made for the existence of temporary edits. If they do exist and their listing is desired, each edit is printed individually. The type of each edit is determined and the required listing routine is called to do the print. The origin of the print request must be determined for, in one instance, the buffer containing the table is split into commands and edits and, in another case, the table is not.

Output: The only output from MDCMTL is the print of the command table.

USAGE

ENTRY MDCMTL
CALL MDCMTL (START,END,WBUF,FLAG)

ARGMT	I/O	TYPE	DIM	DEFINITION
START	I	I	1	THE NUMBER OF THE COMMAND AT WHICH THE PRINT BEGINS.
END	I	I	1	THE NUMBER OF THE COMMAND AT WHICH THE PRINT STOPS
WBUF	I	I	VARB	THE BUFFER CONTAINING THE COMMAND TABLE
FLAG	I	I	1	AN INDICATION OF THE CALLING ROUTINE: #0,CALL BY DUMP DIRECTIVE #1,CALL BY COMMAND TABLE EDITOR #2,DO NOT LIST THE TEMP EDITS

EXTERNAL REFERENCES

MDLSTO
MDLSTI
MDLSTR
MDLSTH

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

COMMON / MDBUFF /
VARB I/O

BDATA I

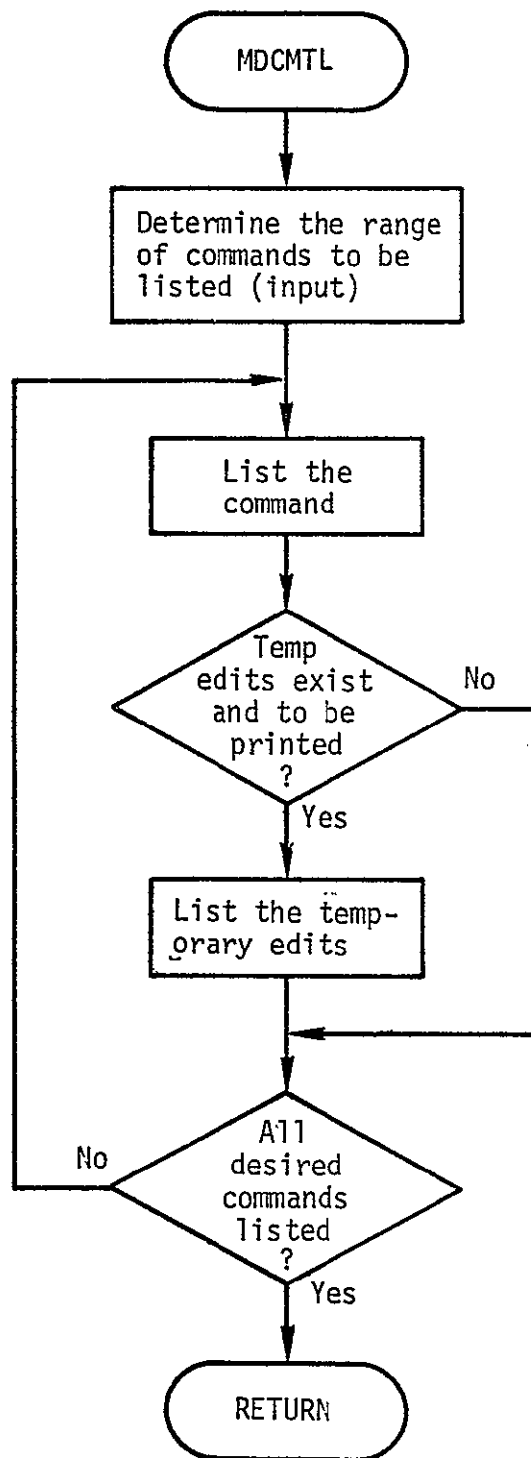
BLANK COMMON

NONE

LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDCMTL Flow Diagram

MDCMTS - Command Table Editor

MDCMTS searches the command table for a requested command (by sequence number) and returns the number of the command corresponding to the sequence number. If the requested number does not match any in the table, the next largest command is returned.

Method

Input: All input is contained in the calling sequence and is: the command table, the number of commands in the table and the sequence number of the requested command.

Processing: If the command table is not empty, it is searched until a sequence number greater than, or equal to, the requested number is found. If none is found the number of the last command in the table is returned. If the found number is not equal to the requested one, a flag is set indicating this and the found command's number is returned.

Output: All output is through the calling sequence and consists of the number of the command corresponding to the requested sequence number, or the number corresponding to the next highest command if the requested number did not exist, and a flag indicating either an empty command table, the requested command existed or the requested command did not exist and the next highest number was returned.

USAGE
ENTRY MDCMTS
CALL MDCMTS (CMDTAB,NCMDS,SEQNO,CMD,STATUS)

ARGMT	I/O	TYPE	DIM	DEFINITION
CMDTAB	I	I	VARB	BUFFER CONTAINING THE COMMAND TABLE
NCMDS	I	I	1	NUMBER OF COMMANDS IN THE TABLE
SEQNO	I	I	1	SEQUENCE NUMBER OF THE DESIRED COMMAND
CMD	O	I	1	NUMBER OF THE COMMAND CORRESPONDING TO SEQNO (OR THE NEXT ONE IF SEQNO IS NOT FOUND)
STATUS	O	I	1	VALIDITY OF STATUS: =0,SEQNO FOUND,RETURN OK =-1,NULL COMMAND TABLE =1,SEQNO NOT FOUND,NEXT COMMAND RETURNED

EXTERNAL REFERENCES
NONE

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE

MDALST - Control Table Editor

The purpose of MDALST is to list or list and prompt the values of a given argument in a control table.

Method

Input: The input to MDALST is the control table, a pointer to the argument specifications in the control table, a pointer to the start of the data in the control table, the length of the control table, and a flag designating whether values are to be output only or output and the response read. This data is passed to MDALST through the calling sequence.

Processing: MDALST calls MDLKUP to locate the argument data for the requested argument. If MDLKUP cannot locate the argument data, then an error message is printed stating the variable is undefined. All messages and displays are output by MDALST for output only requests. For output and read requests, the data is printed and a response prompted by MDPRMT.

Values for "=" (immediate data) and "@" (execution time data) are processed and printed by MDALST or by calls to MDPRMT depending upon the request. Free field data is printed in octal via MDLSTO, real via MDLSTR and integer via MDLSTI if output only is requested; otherwise, MDPRMI is called. Real values are printed by MDLSTR and Hollerith values by MDLSTH, unless the request was for list and read then MDPRMR is called for real data and MDPRMH for Hollerith.

Output: The output from MDALST consists of a buffer containing the user's response as processed by MDSCAN and a status flag. The status flag indicates the type of return from MDALST (0 = normal, other = undefined argument).

USAGE

ENTRY MDALST

CALL MDALST (CTAB,ARGPTR,BDATA,LEN,FLAG,BUFF,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
CTAB	I	I	VARB	CONTROL TABLE
ARGPTR	I	I	1	INDEX INTO THE CONTROL TABLE ARGUMENT
BDATA	I	I	1	SPECS INDEX INTO THE CONTROL TABLE-
LEN	I	I	1	BEGINNING OF THE DATA
FLAG	I	I	1	PROCESSING FLAG.
				TYPE OF PROCESSING
				1=OUTPUT VALUES
				2=OUTPUT VALUES AND READ RESPONSE
BUFF	O	I	VARB	BUFFER CONTAINING USERS RESPONSE AS
				PROCESSED BY MDSCAN
STAT	O	I	1	STATUS OF PROCESSING
				0=NORMAL RETURN
				-1= UP ARROW RESPONSE

EXTERNAL REFERENCES

MDLKUP
MDLSTH
MDLSTI
MDLSTO
MDLSTR
MDPRMH
MDPRMI
MDPRMR
MDPRMT

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

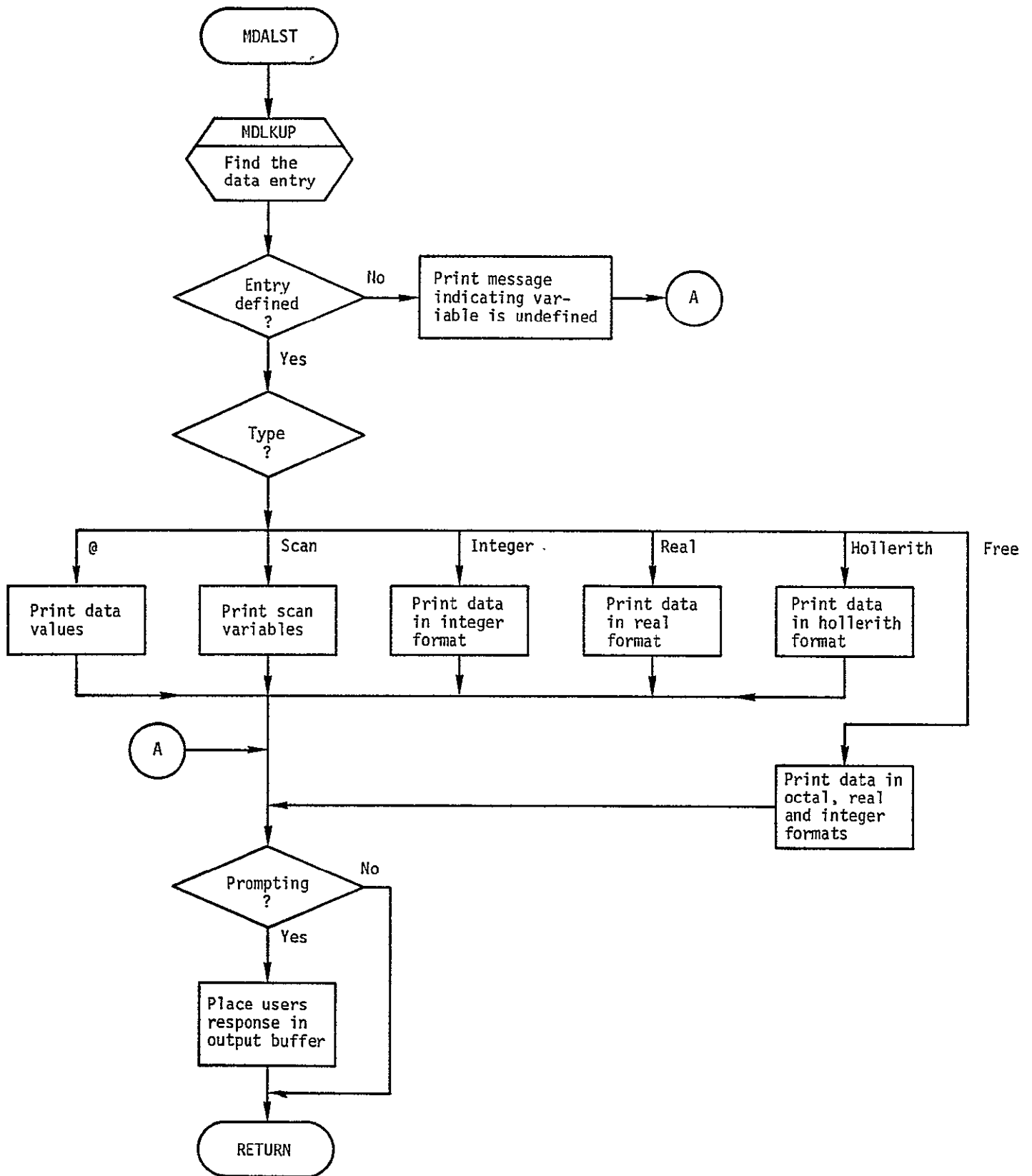
BLANK COMMON

NONE

LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDALST Flow Diagram

MDCNT - Control Table Editor

MDCNT is the primary subroutine of the control table editor. Its purpose is to prompt the user, accept I/O specifications, and place these into a control table. There are two alternate entrances to MDCNT; they are MDEDCN and MDCMTN. MDEDCN is called to complete a control table being used to execute a processor or to modify a control table immediately before execution. MDCMTN is called for temporary edits.

Method

Input: The input to MDCNT is the buffer containing the user's input EDCNT directive after it has been processed by MDSCAN. The EDCNT directive allows the user to interactively create or modify a control table. The control table contains the argument specifications and data needed for the execution of a processor. When the EDCNT directive is requested, MDCNT is called by the submonitor (MDSMON).

Processing

MDCNT must first validate the syntax of the EDCNT directive and then interpret the fields. MDGET is called to bring an existing control table (if specified on the directive) into the working buffer from the SMT. If an existing control table is not specified, MDGETC is called to read the default control table into the working buffer. MDSPLT is called to separate the argument specs and data portions of the control table in the working buffer.

Depending on the option(s) used on the EDCNT directive, the user is prompted only for incomplete arguments or also prompted to concur with existing values of completed arguments. If a "?" is entered to the right of an "=" or "@" sign, MDDEFN is called to list the textual definition of this argument. MDCØNT is called to process all other user responses and to return a status indicating what is to be prompted next.

If the scan available flag of the control table is set and one or more arguments are incomplete, the user is prompted to input &SCANX, &SCANY, and &DATBX. The user may, of course, input or modify any of these

scan control arguments directly.

When the user indicates that no further editing is desired, the complete bit of the control table is set and the control table is packed (by MDCTPK) and stored into the SMT (by MDPUT).

Entry point MDEDCN is called by the execution controller to complete and/or modify a control table about to be used in a processor execution. The control table is already in the working buffer, and split when MDEDCN is called. The control table is not packed and stored into the SMT when MDEDCN is exited.

Entry point MDCMNT is called by the command table editor to build temporary edits. The control table is already in the working buffer and split when MDCMNT is called. The "automatic" prompting loop is not executed for MDCMNT, rather the user specifies all argument to be edited. The control table is not packed and stored into the SMT when MDCMNT is called.

Output: The output from MDCNT, MDEDCN, and MDCMTM consists of a control table either created or modified and a flag indicating the status of the routines processing. Since many routines are called, a negative status will be set by the routine encountering an error; unless a fatal error occurs, then the control table editor will set the status flag indicating this error.

Word 1	Processor Name																																		
Word 2	Revision #, 0 7		# of Arguments 8 12		COMP 13	SCBL 14	SCAN 15	Not Used 16 35																											
Arg. Spec. Entry	Argument Identifier (alphanumerical name)																																		
	I-Dim 0 7							J-Dim 8 15							Type 16 27							C 28	I/O 29 30		CF 31	P 32	I 33	34 35							
	.																																		
	.																																		
Data Entry	Label (Arg. Number) 0 17																	Size 18 25																	
	Data for this Arg.																																		
	Label 0 17																	Size 18 35																	
	Data																																		
	.																																		
	.																																		
	.																																		

Control Table

Control Table Definitions

Header (first 2 words)

of Arguments: # of arguments in this control table (31)
COMP (Bit 13): Is complete data specified for all arguments
 = 0, complete
 = 1, incomplete
SCBL (Bit 14): Is scan permitted
 = 0, No
 = 1, Yes
SCON (Bit 15): Is scan active
 = 0, No
 = 1, Yes

Argument specifications

Total of 2* # of arguments words. If scan is active 6 additional words of argument specifications exist.

I-Dim: I-dimension of this argument

J-Dim: J-dimension.

Type: 2000 = user local free
 2001 = user local integer
 2002 = user local real
 etc.

C: Constant flag
 = 0, Normal arg.
 = 1, Constant arg.
 = 2, Scan variable
 = 3, Data box

I/O: I/O flag
 = 0, Input
 = 1, Input/output
 = 2, Output

CF: Completion flag
 = 0, Argument is currently undefined
 = 1, Incomplete
 = 2, Complete

P: Print flag
 = 0, Argument to not be printed at execution time
 = 1, Printed

I: Indication
 = 0, Immediate data (=)
 = 1, Execution time data (@)

Data (remainder of table consists of variable length entries)

Label: Argument number (relative to first arg.)

Size: # of words (including this header) contained in this data entry

Data: for '=' these are SIZE-1 values
 for '@' this is 'NAME' (name of SMT entry where data exists),
 type, (2000, 2001, etc.), I-Sub, J-Sub

NOTE: In the working buffer, the data area is filled from the bottom up.
 If an argument appears in the data area more than once the data located
 "highest" in the data area is used.

USAGE

ENTRY MDCNT

CALL MDCNT (BUFF,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
BUFF	I	I	VAR8	INPUT BUFFER WHICH HAS BEEN PROCESSED BY MDSCAN
STAT	O	I	1	STATUS OF MDCNT PROCESSING 0=NORMAL RETURN -1=FATAL ERROR NEG=ERROR STATUS RETURNED FROM OTHER SUBROUTINES CALLED

EXTERNAL REFERENCES

MDALST
MDCNTS
MDCONT
MDCTPK
MDDEFN
MDGET
MDGETC
MDIMS
MDLIST
MDLKUP
MDPRMT
MDPUT
MDSPEC
MDSPLT
SEARCH

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

DIAGNOSTICS

*** REVISION NUMBER OF PROCESSOR (...) DOES NOT MATCH
THAT OF CONTROL TABLE (...).
THE PROCESSOR'S REVISION NUMBER DOES NOT MATCH
THE REVISION NUMBER IN THE CONTROL TABLE.
*** WARNING *** IS A CONTROL TABLE FOR
BUT THIS EDIT WILL PRODUCE A CONTROL TABLE
FOR
A CONTROL TABLE WAS REQUESTED FOR AN INCORRECT
PROCESSOR
*** CONFLICT BETWEEN TYPE OF DATA INPUT AND TYPE OF
INCORRECT DATA SPECIFIED
*** SUBSCRIPT OUT OF RANGE -- DIMENSION OF IS,
.....
SUBSCRIPT OUTSIDE RANGE
*** SYNTAX ERROR -- TRY AGAIN
SYNTAX ERROR FOUND
*** COULD NOT FIND NAME IN LIST -- TRY AGAIN !
DATA FOR NAME COULD NOT BE FOUND IN IMS
*** TOO MUCH DATA FOR ONLY ACCEPTED, ENOUGH TO FILL
ARRAY
TOO MUCH DATA WAS INPUT
*** NOT A LEGAL ARGID FOR PROCESSOR,
INCORRECT ARGUMENT ID SPECIFIED FOR PROCESSOR
*** READ ERROR IN READING FROM ON LINE, STORAGE DEVICE
READ ERROR OCCURRED FROM RAD
*** MAY NOT OUTPUT A GLOBAL IMS (\$) VARIABLE

AN IMS VARIABLE WAS REQUESTED FOR OUTPUT
 *** MAY ONLY BE SPECIFIED WITH =
 AN OUTPUT VARIABLE WAS SPECIFIED AS AN INPUT VARIABLE

F A T A L E R R O R S

*** INVALID SYNTAX ***
 AN INVALID SYNTAX WAS SPECIFIED
 *** INVALID PROCESSOR NAME ***
 THE PROCESSOR NAME SPECIFIED WAS NOT FOUND IN PROTAB
 *** TOO LARGE FOR WORKING BUFFER
 THE CONTROL TABLE FOR PROCESSOR IS TOO LARGE
 FOR THE WORKING BUFFER.
 *** READ ERROR IN MDGET ***
 READ ERROR OCCURRED IN MDGET
 *** COULD NOT FIND IN IMS ***
 COULD NOT FIND DATA FOR PROCESSOR NAMED IN IMS
 *** COULD NOT FIND IN SMT ***
 COULD NOT FIND DATA FOR PROCESSOR NAMED IN SMT
 *** COULD NOT FIND DEFAULT CONTROL TABLE FOR
 DEFAULT CONTROL TABLE FOR COULD NOT BE FOUND
 BY MDGETC
 *** READ ERROR TRYING TO READ DEFAULT, CONTROL TABLE
 A READ ERROR OCCURRED WHILE TRYING TO READ A DEFAULT
 CONTROL TABLE
 *** I/O ERROR WHILE FORMATTING A PROMPT
 AN I/O ERROR OCCURRED WHILE FORMATTING A PROMPT
 *** READ ERROR WHILE READING RESPONSE
 A READ ERROR OCCURRED WHILE READING A RESPONSE FROM
 MDPRMT
 *** WORKING BUFFER OVERFLOW ***
 CONTROL TABLE WORKING BUFFER NOT LARGE ENOUGH TO
 HOLD DATA
 *** UNIDENTIFIABLE STATUS FROM, MDCONT
 AN UNIDENTIFIABLE STATUS VALUE WAS RECEIVED FROM
 MDCONT
 *** ERROR WHILE WRITING TO ONLINE STORAGE
 RAD WRITE FAILED
 *** SUB-MONITOR TABLE (SMT) FULL ***
 COULD NOT ENTER SMT ENTRY

EXTERNAL STORAGE
 NONE

BLANK COMMON
 VARB I/O

PROTAB I
 PTBLLEN I

COMMON /MDCODE/
 ASTRSK I
 AT I
 BCKSLH I
 COMMA I
 DOLLAR I
 EOS I
 EQUALS I
 LPAR I

NAME	I				
PERCNT	I				
QUESMK	I				
RPAR	I				
SUBS	I				
UPARRW	I				

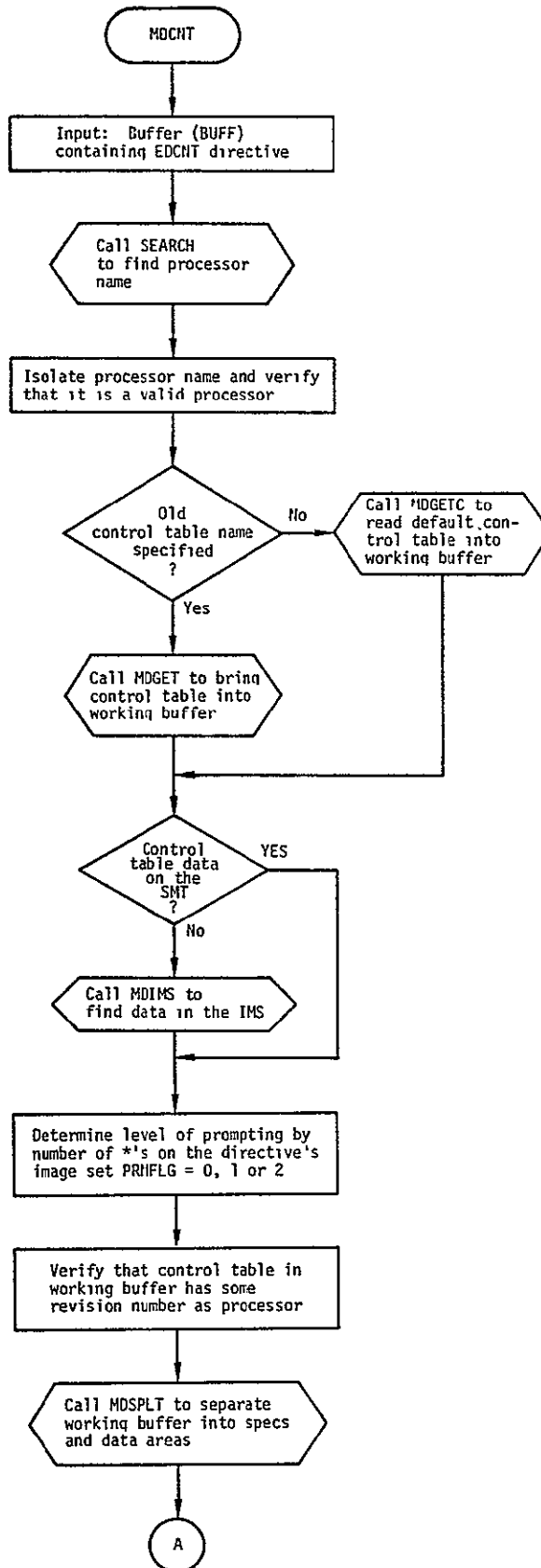
COMMON /MDBUFF/

RDATA	I/O				
MDLEN	I				
SIZE	I/O				
WB	I/O				

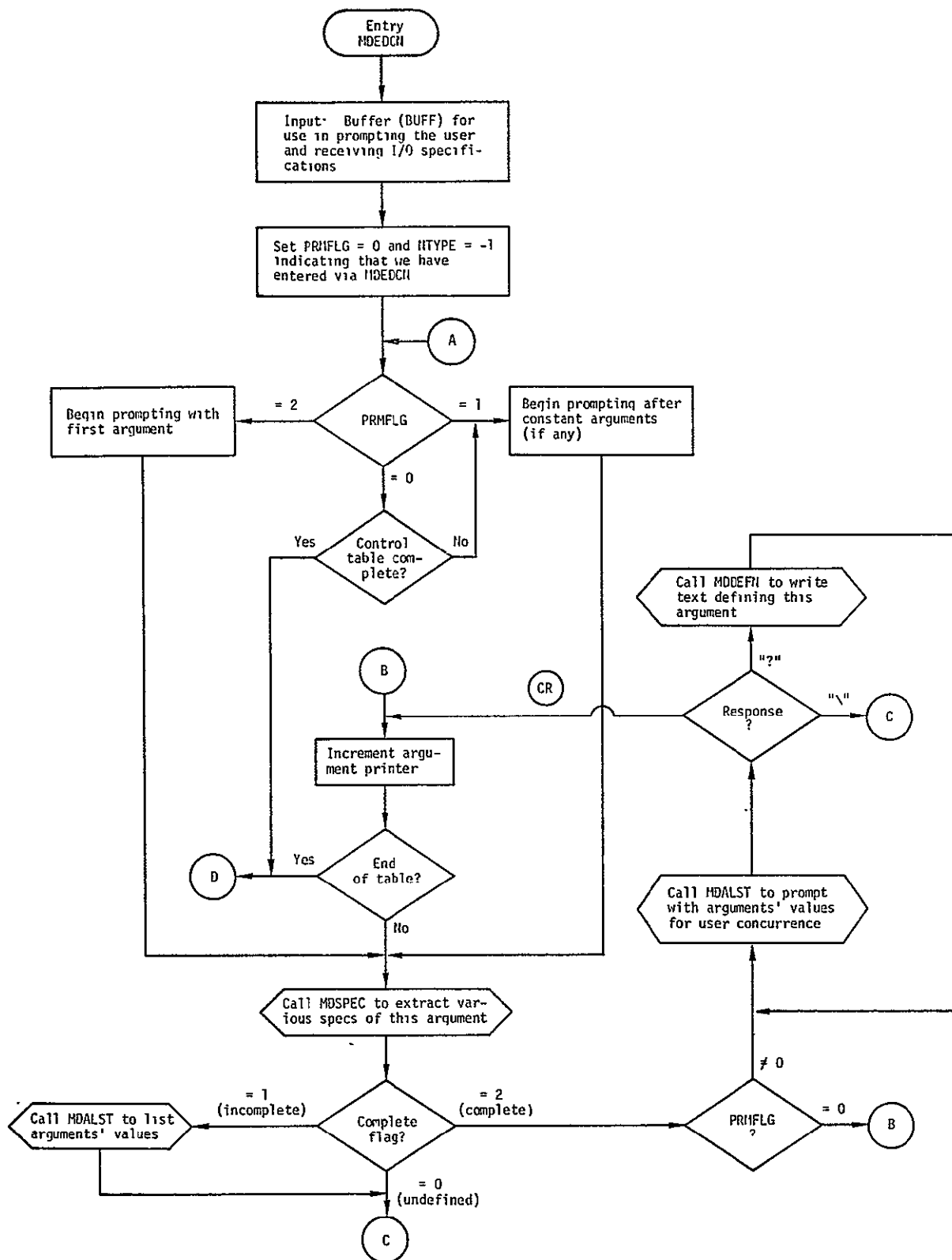
LOCAL COMMON

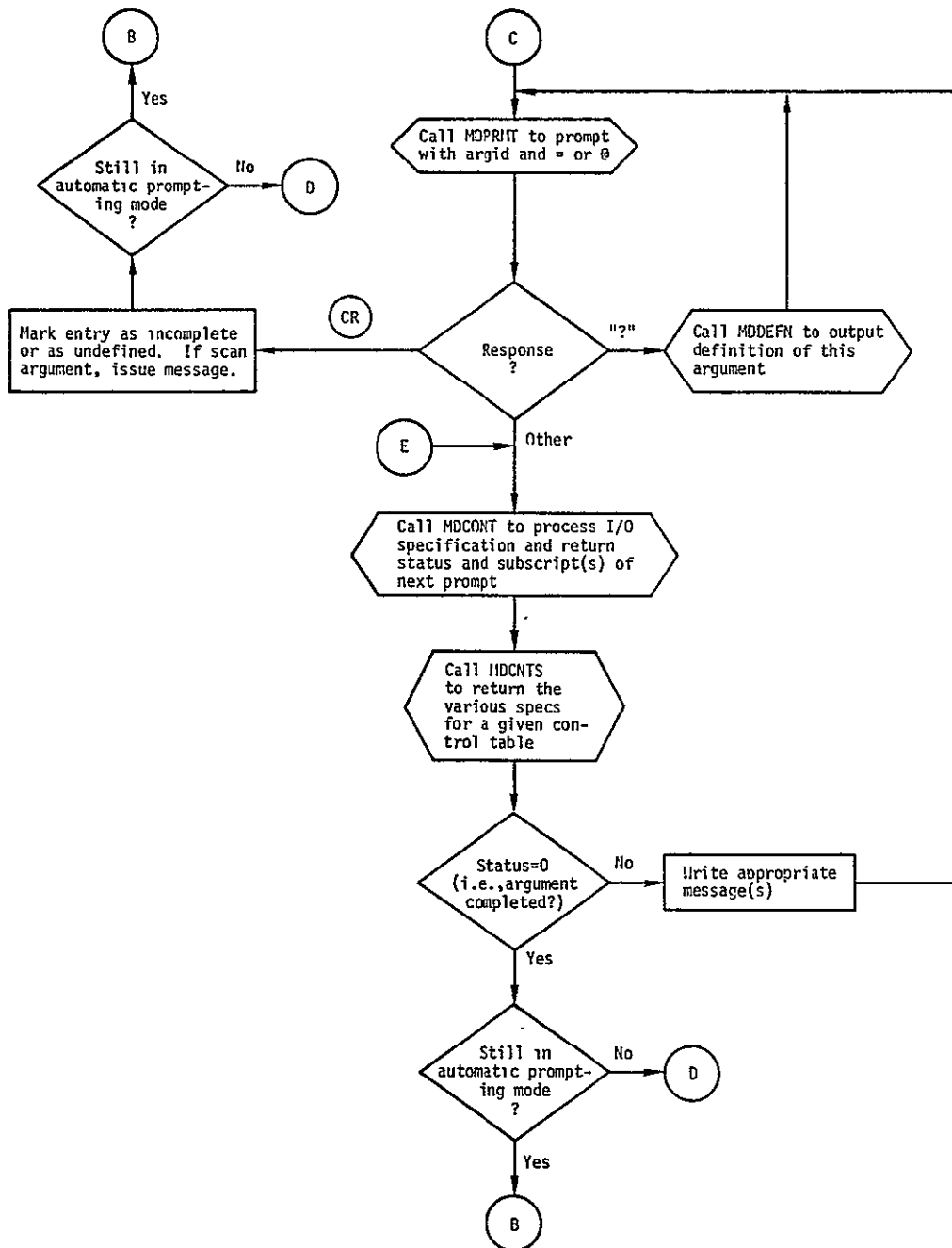
VARB	I/O	TYPE	DIM	LOC	RELADD	DEFINITION
------	-----	------	-----	-----	--------	------------

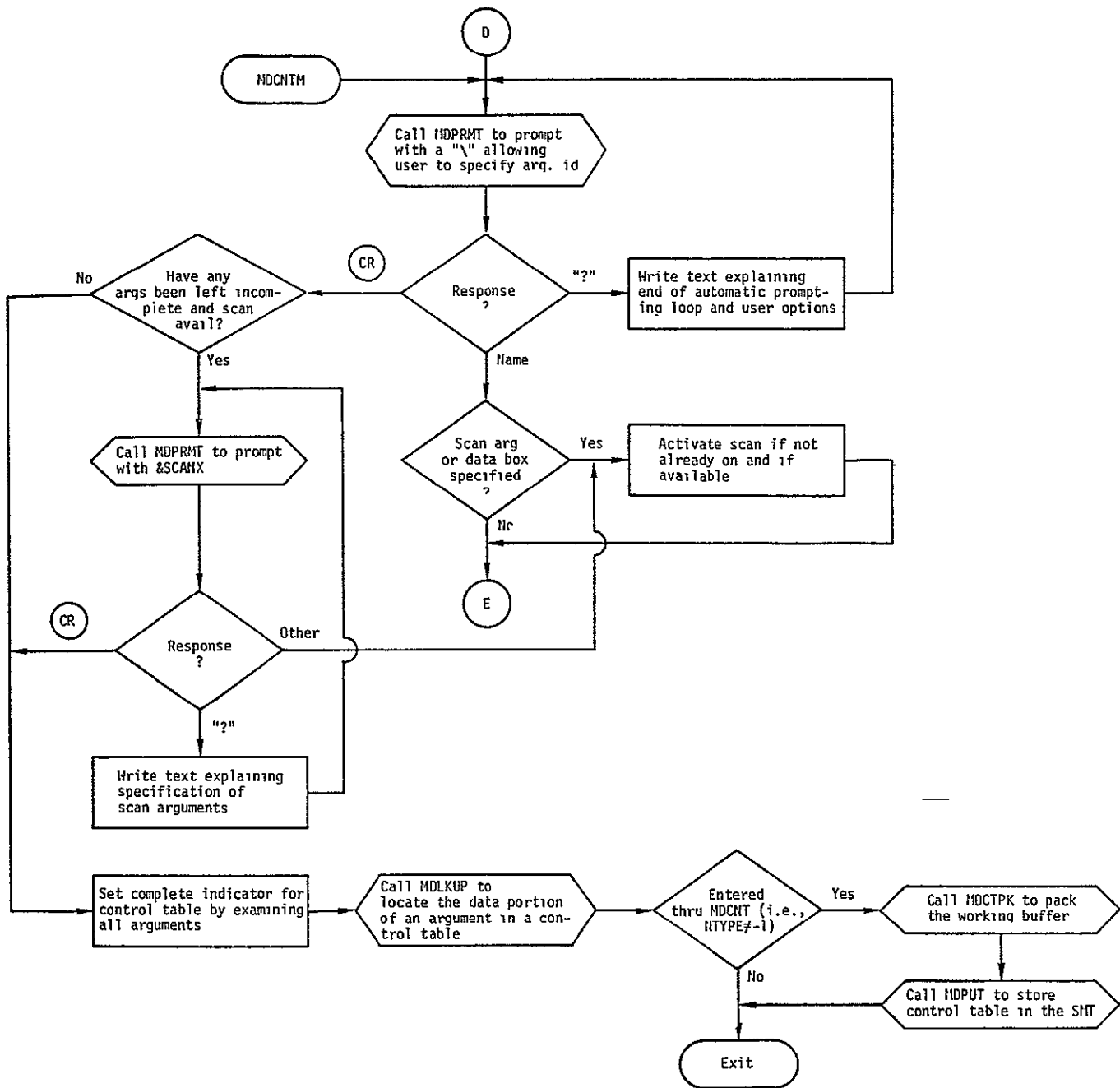
NONE



EDCNT Flow Diagram







MDCNT Flow Diagram (Continued)
7.2-12

MDCNTE - Control Table Editor

The purpose of MDCNTE is to process the values to the right of an equals sign. It transfers the data values from the input buffer into the control table contained in the working buffer. MDCNTA is an alternate entrance and has the same purpose as MDCNTE except it processes value to the right of an at (@) sign.

Method

Input: The input to MDCNTE consists of the following:

- (a) The control table in the working buffer and the index to the data entry to be filled.
- (b) The user's edit line as output by MDSCAN and the index to the next field in this buffer.
- (c) The displacement into this data entry at which the first field belongs.
- (d) The first and second subscript limits.
- (e) The argument completion status.
- (f) The type of argument from argument specifications.

The input to MDCNTA is the same input as (a), (b), and (f) of MDCNTE plus a flag designating the I/O for the argument.

Processing: MDCNTE will update a control table with the values to the right of an equals on a control table edit. Validity checks will be made and when an error condition occurs the status flag is set accordingly.

It is verified that values (real, integer, Hollerith or octal) requested have the correct data type specified, that inserted data may not overflow the table, and that when a % or \$ is requested a name must follow. It is also verified that subscripts are within their defined limits. There is an exception to this test. If the subscript designates the array to be used as a vector and the user has reversed the subscripts then MDCNTE will allow the subscript to be processed.

When a variable name is specified, MDGET is called to get the data from the SMT. If the data is not found in the SMT, MDIMS is called to find the data in the IMS. If the data exists in neither area, an error condition is flagged.

There is an alternate entrance, MDCNTA, which updates a control table with fields on the right side of an at (@) sign on a control table edit. It sets the data type flag to indicate temporary, user's permanent, or IMS data into the control table. It also stores the subscripts into the control table in the working buffer.

Whether the program was entered from MDCNTE or MDCNTA, a check for a valid end-of-statement is made and the output data flags set before the subroutine returns to the caller.

Output: The output from MDCNTE(A) consists of an asterisk status flag, a termination indicator, a counter containing the origin displacement from the first data word to the last data word filled, and a flag indicating the processing status.

USAGE

ENTRY MDCNTE

CALL MDCNTE (CTAB,CPTR,BUFF,BPTR,DISP,IDIM,JDIM,COMP,TYPE,ASTAT,TERM,COUNT,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
CTAB	I	I	VARB	CONTROL TABLE
CPTR	I	I	I	INDEX TO THE DATA ENTRY TO BE FILLED
BUFF	I	I	VARB	USER'S EDIT LINE AS OUTPUT BY MDSCAN
BPTR	I	I	I	INDEX TO NEXT FIELD IN BUFF
DISP	I	I	I	DISPLACEMENT INTO THIS DATA ENTRY AT WHICH FIRST FIELD BELONGS
IDIM	I	I	I	I-DIMENSION, 1ST SUBSCRIPT LIMIT
JDIM	I	I	I	J-DIMENSION, 2ND SUBSCRIPT LIMIT (IF SINGULARLY DIMENSIONED, THIS IS THE SUBSCRIPT LIMIT)
COMP	I	I	I	COMPLETION FLAG 2=ARGUMENT COMPLETE NOT EQUAL TO 2=NOT CURRENTLY COMPLETE
TYPE	I	I	I	TYPE OF THIS ARGUMENT (FROM ARGUMENT SPECS.)
ASTAT	O	I	I	ASTERISK STATUS 0=NO ASTERISK, 1=ONE *, 2=TWO *'S
TERM	O	I	I	TERMINATION INDICATOR 0=NO BACKSLASH 1=LINE TERMINATED WITH BACKSLASH
COUNT	O	I	I	ONE ORIGIN DISPLACEMENT FROM THE 1ST DATA WORD TO THE LAST DATA WORD FILLED.
STAT	O	I	I	RETURN STATUS 1=SIZE OF DATA IS GREATER THAN THE MAXIMUM SIZE 0=ARGUMENT COMPLETE - 2=INCORRECT DATA TYPE - 3=TOO MUCH DATA INPUT - 4=SUBSCRIPT OUT OF RANGE - 5=SYNTAX ERROR - 9=READ ERROR FROM RAD -11 DATA NOT IN IMS

ENTRY MDCNTA

CALL MDCNTA (CTAB,CPTR,BUFF,BPTR,IOFLG,TYPE,ASTAT,TERM,COUNT,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
CTAB	I	I	VARB	CONTROL TABLE
CPTR	I	I	I	INDEX TO THE DATA ENTRY TO BE FILLED
BUFF	I	I	VARB	USER'S EDIT LINE AS OUTPUT BY MDSCAN
BPTR	I	I	I	INDEX TO NEXT FIELD IN BUFF
IOFLAG	I	I	I	INPUT/OUTPUT FLAG 0=ARGUMENT IS INPUT 1=ARGUMENT IS INPUT/OUTPUT 2=ARGUMENT IS OUTPUT
TYPE	I	I	I	TYPE OF THIS ARGUMENT (FROM ARGUMENT SPECS.)
ASTAT	O	I	I	ASTERISK STATUS

TERM	0	I	1	0=NO ASTERISK,1=ONE *,2=TWO **S TERMINATION INDICATOR
COUNT	0	I	1	0=NO BACKSLASH 1=LINE TERMINATED WITH BACKSLASH ONE ORIGIN DISPLACEMENT FROM THE 1ST DATA WORD TO THE LAST DATA WORD FILLED.
STAT	0	I	1	RETURN STATUS 1=SIZE OF DATA IS GREATER THAN THE MAXIMUM SIZE 0=AUGUMENT COMPLETE - 5=SYNTAX ERROR -10=NO DATA INPUT-ARGUMENT IS TO BE MARKED UNDEFINED -12=MAY NOT OUTPUT AN IMS VARIABLE GT 0=INCOMPLETE--COUNT+1 IS THE NEXT WORD OF THE ARRAY TO BE FILLED

EXTERNAL REFERENCES

MDGET
MDIMS

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

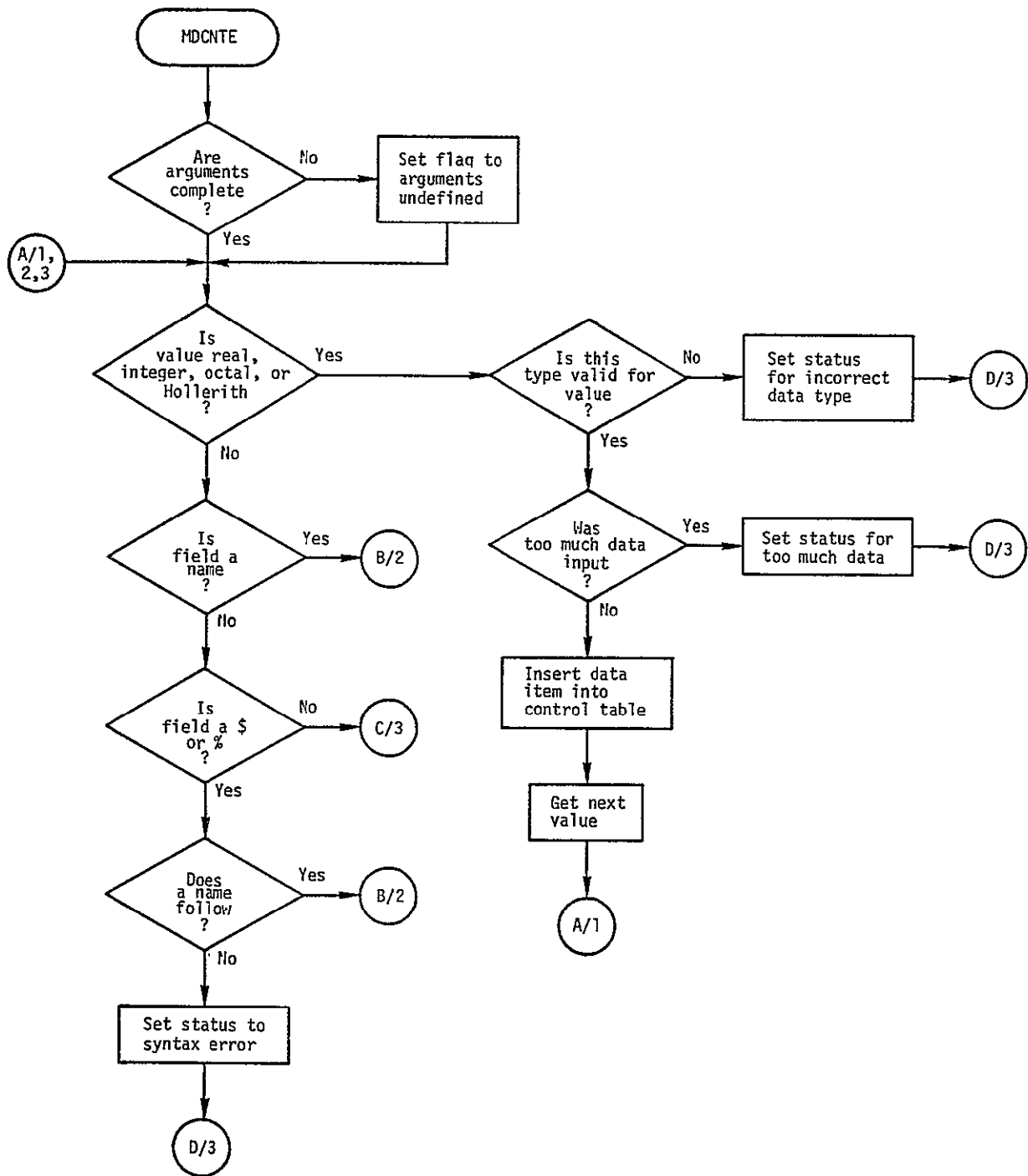
NONE

COMMON /MDCODE/

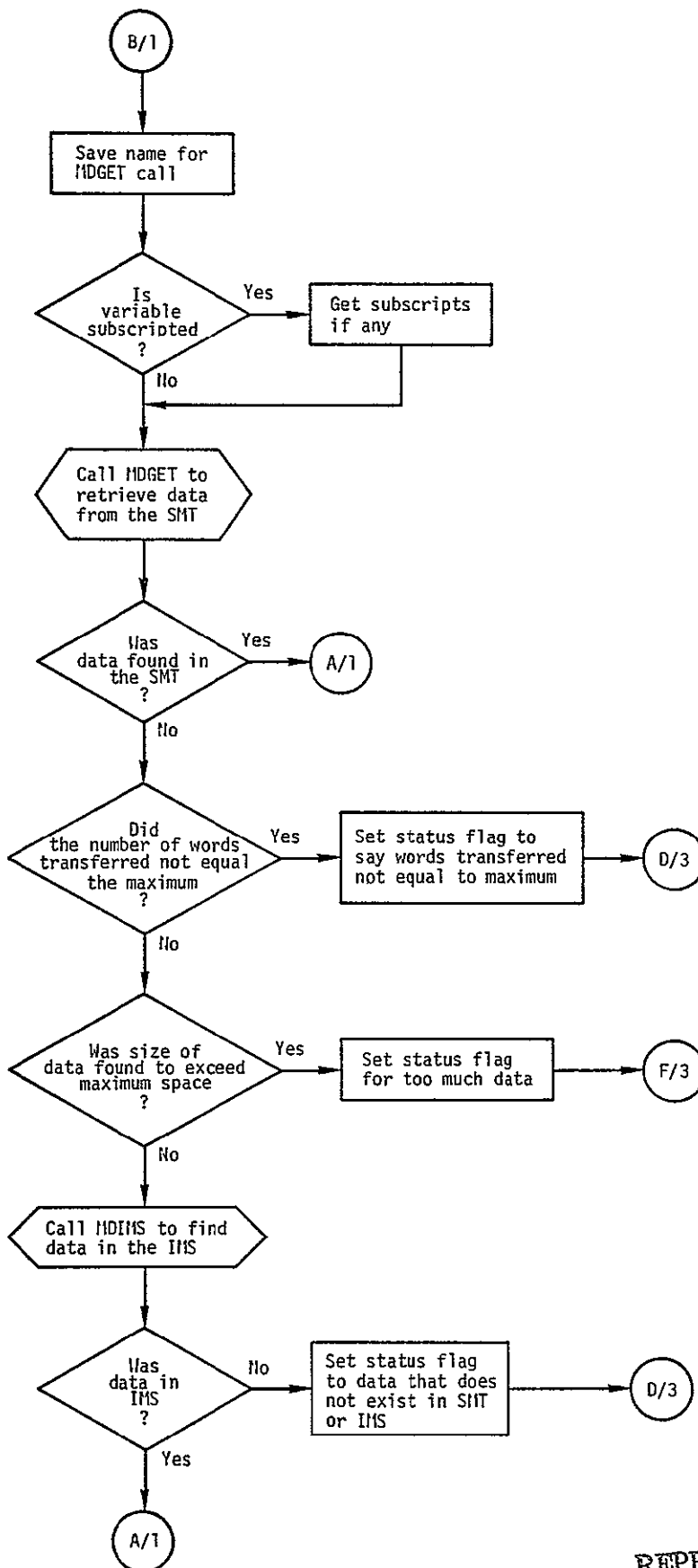
VARB	I/O
ASTRSK	I
BCKSLH	I
DOLLAR	I
EOS	I
HOLLRH	I
INTEGR	I
NAME	I
OCTAL	I
PERCNT	I
REAL	I
REPEAT	I
SUBS	I

LOCAL COMMON

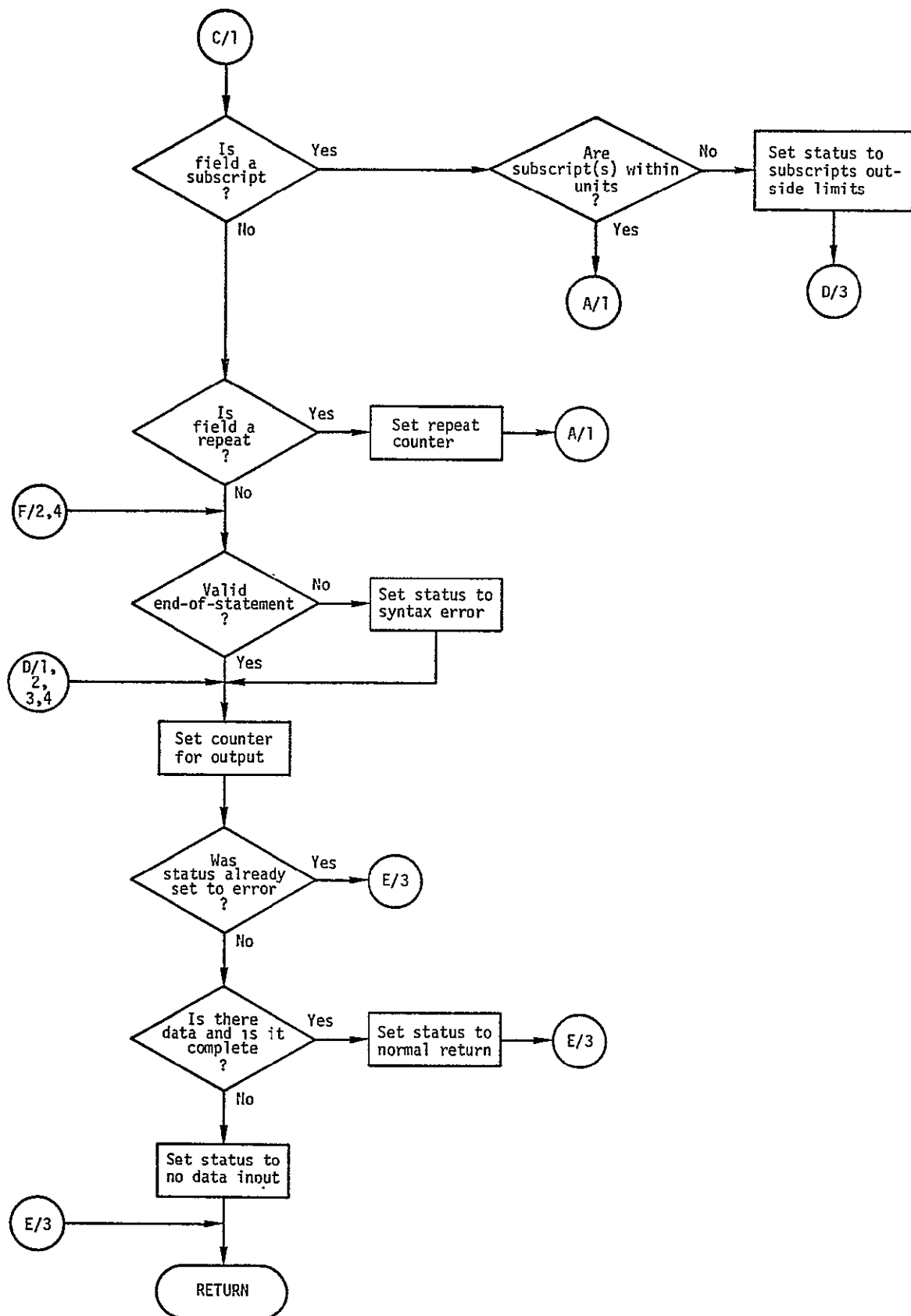
NONE

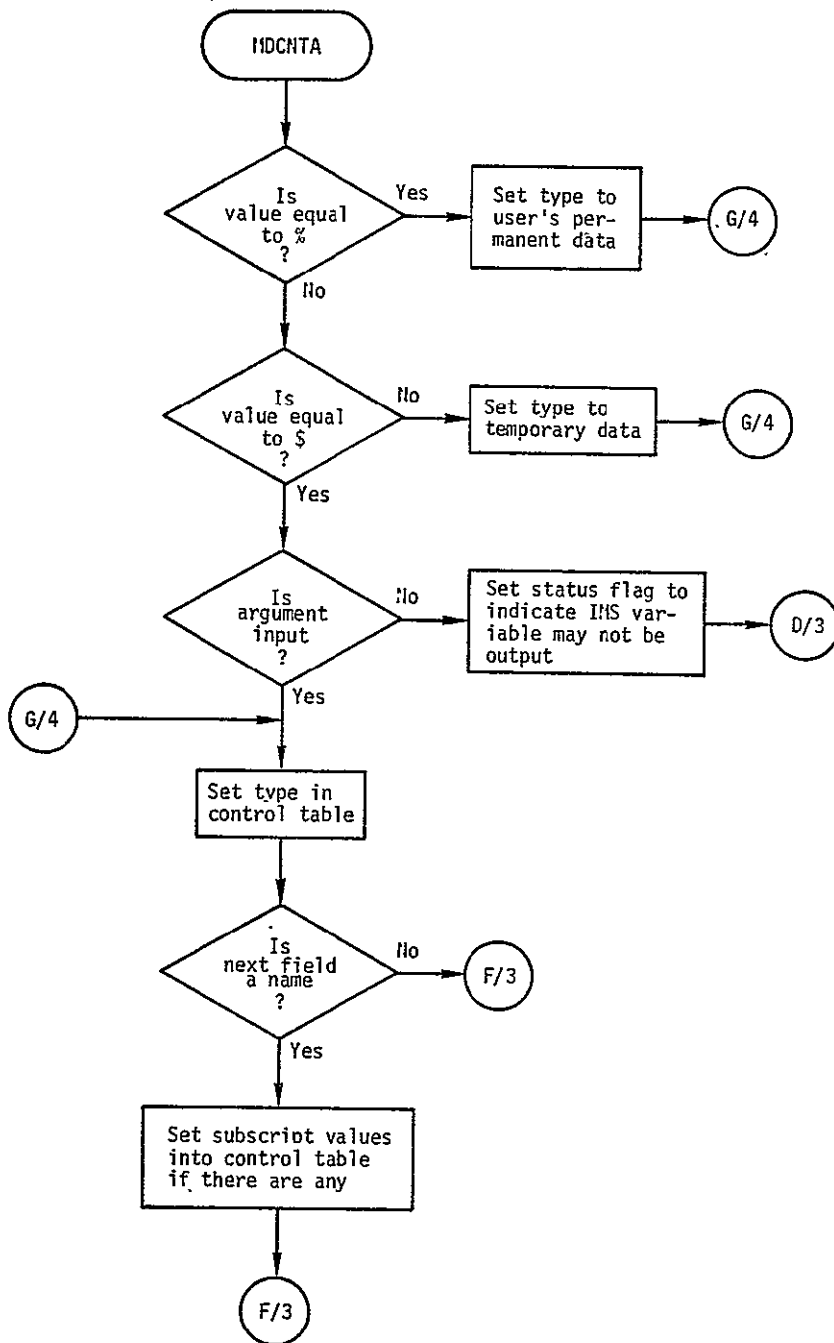


MDCNTE Flow Diagram
7.3-5



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR





MDCNTS - Control Table Editor

The purpose of MDCNTS is to search a given control table for an argument identifier and return its argument specifications.

Method

Input: The input to MDCNTS consists of an argument identifier and control table.

Processing: MDCNTS searches the control table until an argument identifier match is found. The entry number of this identifier is saved as the argument number. The argument number and the control table are passed to MDSPEC to get the output parameters.

Output: The output from MDCNTS consists of the following control table items:

- Argument number
- Type of variable
- I-dimension (1st subscript limit)
- J-dimension (2nd subscript limit)
- I/O flag
- Completion flag
- @ indicator flag

For more detailed information about the control table refer to MDCNT.

USAGE

ENTRY MDCNTS

CALL MDCNTS (ARGID,CTAB,ARGNUM,TYPE,IDIM,JDIM,IOFLG,COMPL,
EQUAT,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
ARGID	I	H	1	ARGUMENT IDENTIFICATION
CTAB	I	I	VARB	CONTROL TABLE
ARGNUM	O	I	1	ARGUMENT NUMBER
TYPE	O	I	1	ARGUMENT TYPE
IDIM	O	I	1	I-DIMENSION, 1ST SUBSCRIPT LIMIT
JDIM	O	I	1	J-DIMENSION, 2ND SUBSCRIPT LIMIT
IOFLG	O	I	1	INPUT/OUTPUT FLAG 0=INPUT 1=INPUT/OUTPUT 2=OUTPUT
COMPL	O	I	1	COMPLETION FLAG 0=ARGUMENT IS CURRENTLY UNDEFINED 1=THIS ARGUMENT IS INCOMPLETE 2=THIS ARGUMENT IS COMPLETE
EQUAT	O	I	1	EQUAL/AT FLAG 0=DATA FOR THIS ARGUMENT ARE IMMEDIATE VALUES (=) 1=THE DATA FOR THIS ARGUMENT ARE TO BE DETERMINED AT RUN TIME (AT)
STAT	O	I	1	STATUS FLAG 0=NORMAL RETURN -1=COULD NOT FIND ARGUMENT I.D.

EXTERNAL REFERENCES
MDSPEC

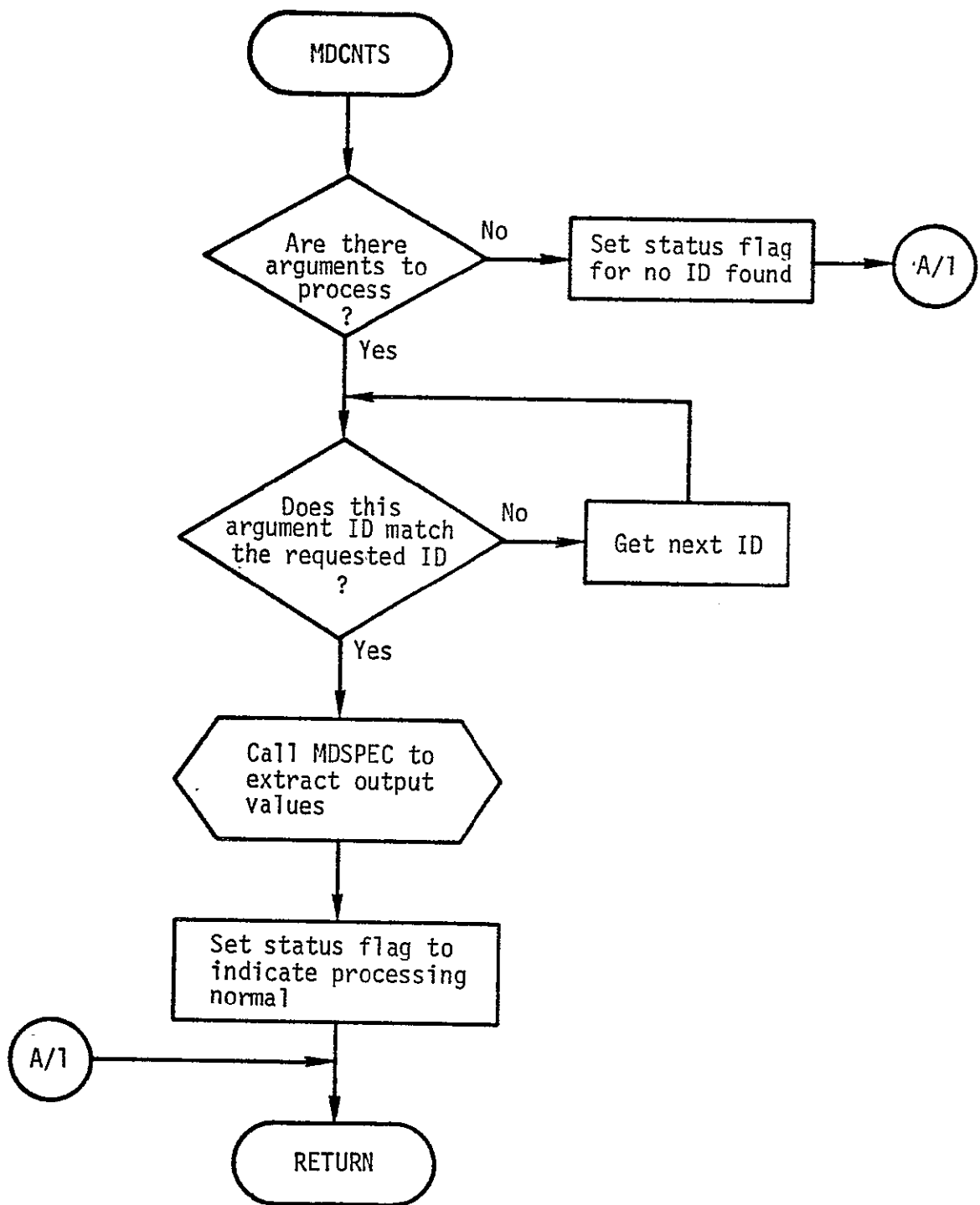
DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDCNTS Flow Diagram

MDCONT - Control Table Editor

The purpose of MDCONT is to process one user generated control table edit.

Method

Input: The input to MDCONT consists of the user input buffer after MDSCAN's processing.

Processing: MDCONT sets the completion flag and print flag in the argument specifications and the label (argument number) and size of data entry in the data area of the control table. The argument number is found by calling MDEDIT to locate the argument ID and then using this ID as input to MDCNTS to locate the argument number.

The size of the data entry is determined when the argument ID has been previously undefined or the = \@ flag has changed. When this condition occurs, the data entry size is calculated in two ways. If the request is for an equals, the size is determined by the product of the I and J dimensions plus one. If the request were for an @ sign, the argument size is set to five. For an @ sign request it requires one word each for the name, type, I-dimension, J-dimension, and the label. If the argument ID is defined and the = \@ flag has not changed then MDLKUP is called to look up the data portion of an argument in the control table.

MDCONT verifies that subscripts are within the maximum size and correctly used. It also verifies that neither an output argument nor RAD resident data is specified with an equals. After the validation checks have been passed, MDCNTE (MDCNTA) is called to update the control table with values on the right side of an equals sign (at sign).

MDCONT also processes print requests. If arguments from the control table are requested to be listed (denoted by *), MDALST is called to list them. If the arguments are to also be listed at execution time (denoted by **), then the print flag is set in the control table.

After the data has been verified, the completion flag is set in the argument specifications of the control table. The setting of the completion flag is determined from the status returned from the other subroutines called. The I and J dimensions are also set into the control table. When processing is complete a status flag is returned containing the conditions found during processing.

Output: The output from MDCONT consists of a flag containing the processing status and the I and J dimensions for subscripts.

USAGE

ENTRY MDCONT

CALL MDCONT (BUFF, IDIM, JDIM, STATUS)

ARGMT I/O TYPE DIM

DEFINITION

BUFF I I VARB

THE USER INPUT BUFFER AFTER MDSCAN PROCESSING

STATUS 0 I I

STATUS FROM MDCONT PROCESSING

IDIM 0 I I

I-DIMENSION, 1ST SUBSCRIPT LIMIT

JDIM 0 I I

J-DIMENSION, 2ND SUBSCRIPT LIMIT

0=ENTRY COMPLETE

1=ENTRY COMPLETE BUT USER REQUESTED ADDITIONAL OPPORTUNITY FOR INPUT

2=DATA OF INCORRECT TYPE FOUND PROMPT WITH 'IDIM' AND 'JDIM'

3=TOO MUCH DATA INPUT

4=SUBSCRIPT OUT OF RANGE

5=SYNTAX ERROR, PROMPT WITH 'IDIM' AND 'JDIM'

6=INCOMPLETE AND SHOULD PROMPT FOR ADDITIONAL VALUES AT 'IDIM' AND 'JDIM' SUBSCRIPTS

7=INVALID ARGUMENT ID

8=WORKING BUFFER OVERFLOW

10=ARGUMENT TO BE MARKED AS UNDEFINED

11=COULD NOT FIND NAME GIVEN IN LIST

14=ARGUMENT MARKED AS INCOMPLETE

15=RAD RESIDENT SPECIFIED WITH

16=ONLY * INPUT, REPROMPT ARGUMENT

EXTERNAL REFERENCES

MDALST

MDCNTA

MDCNTE

MDCNTS

MDEDIT

MDLKUP

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

NONE

COMMON /MDBUFF/

MDLEN I

BDATA I/O

WB I/O

COMMON /MDCODE/

EOS I

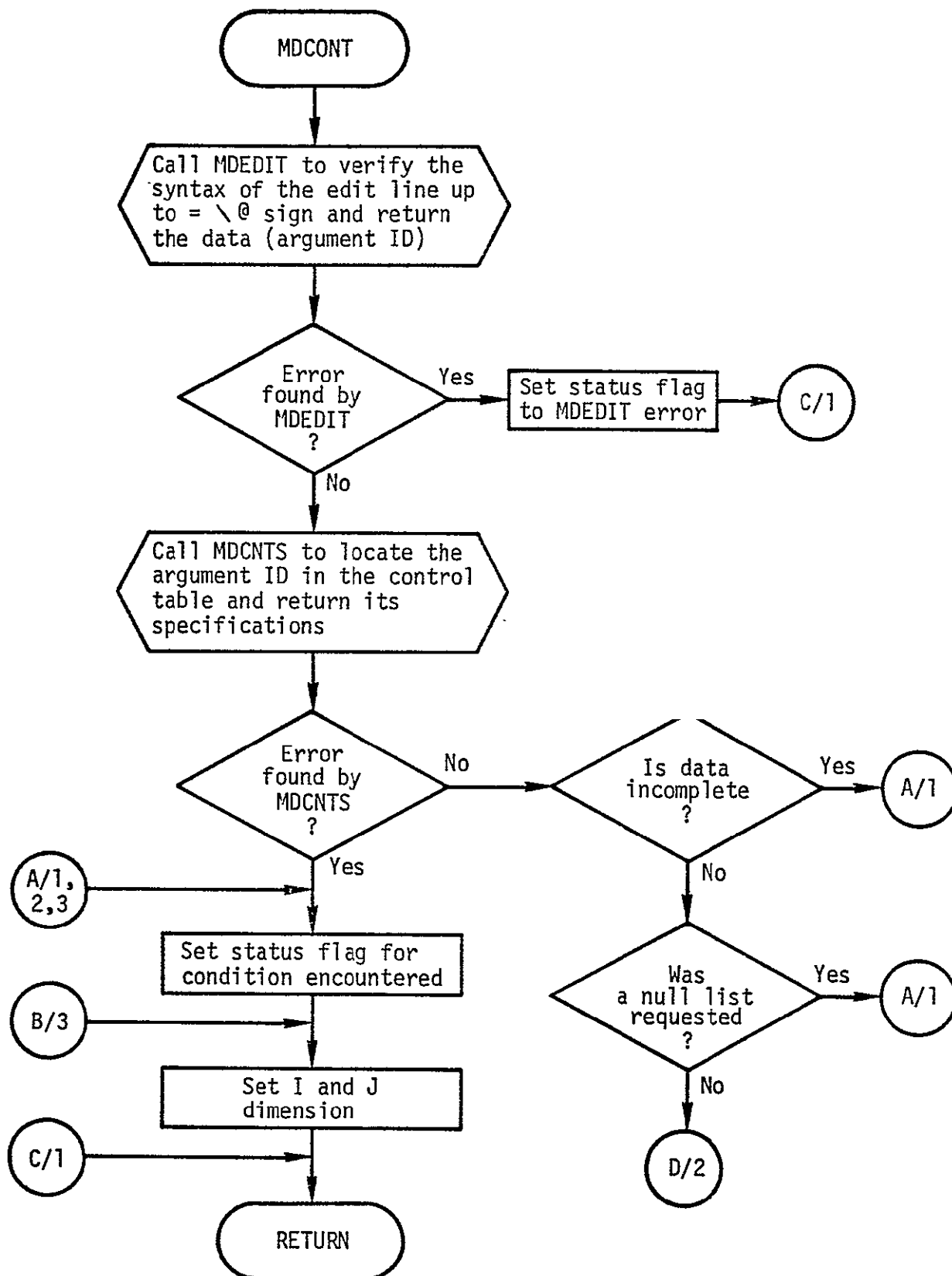
ASTRSK I

BACKSL I

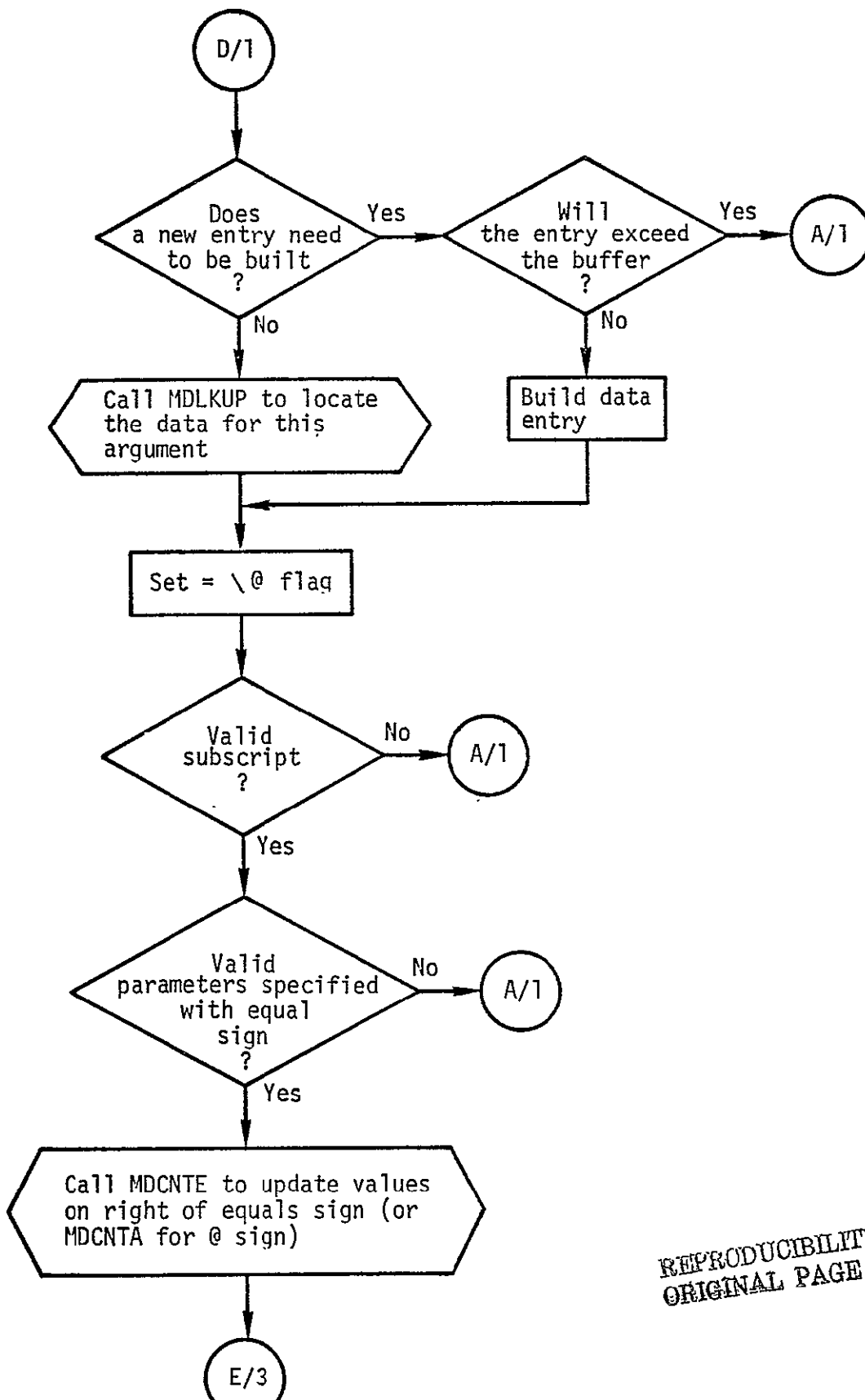
REPRODUCIBILITY OF THE ORIGINAL PAGE IS POOR

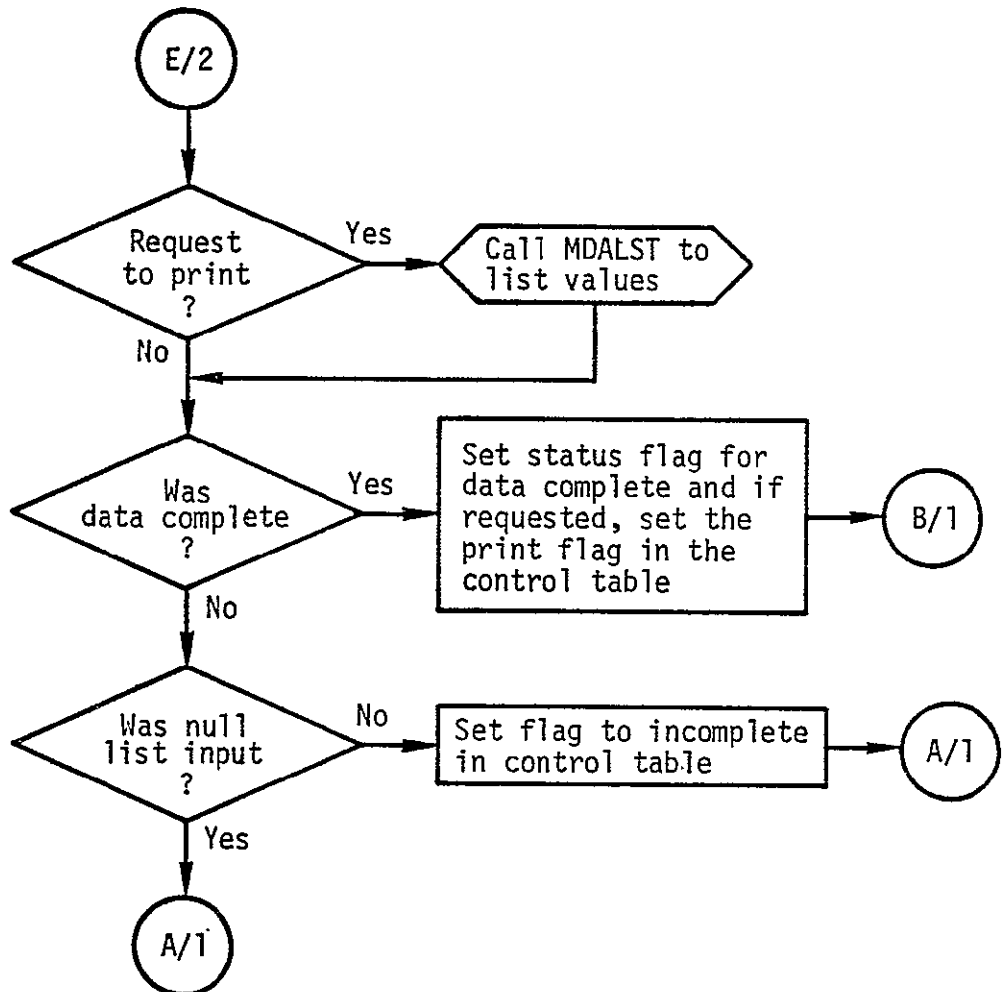
LOCAL COMMON

NONE



MDCONT Flow Diagram





MDDEFN - Control Table Editor

MDDEFN, when developed and implemented, will support the "?" feature of the control level syntax. This feature allows the user to receive an on-line definition of any argument for the processor being edited. These definitions are intended to be kept in an organized RAD data base for quick access.

MDEDIT -- Control Table Editor

The purpose of MDEDIT is to process the left half of an I/O specification (i.e., up through the = or the @) and check its syntax.

Method

Input: The input to MDEDIT is the user's input buffer after MDSCAN processing and an index pointing to the beginning location in this buffer from where processing is to begin. These values are passed through the calling sequence.

Processing: MDEDIT verifies the order and sequence of the parameters for the argument identification, subscripts (I-Dimension and/or J-Dimension), "=", and "@" values. The argument identification must follow a "\"; if it does not, then it must be the first parameter in the buffer. Any condition other than the above, is flagged as an error.

Single or double subscripts are valid but they may only be followed by "=". An "=" means the value is the immediate value following equals. If there were no subscript parameters specified, an "=" or "@" is valid. An "@" means the value will be determined at execution time. If any other combinations occur (i.e., an "@" after a subscripted value), they will be flagged as errors.

Output: The output from MDEDIT consists of the argument identifier, subscript(s) (if any), =/@ flag, index to the next field in the buffer, and status flag for its processing. These parameters are returned through the calling sequence.

USAGE

ENTRY MDEDIT

CALL MDEDIT (BUFF,BPTR,ARGID,SUB1,SUB2,EQUAT,STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
BUFF	I	I	VARB	USER'S INPUT BUFFER AFTER MDSCAN PROCESSING
BPTR	I	I	I	INDEX INTO BUFF
ARGID	O	H	I	ARGUMENT IDENTIFIER
SUB1	O	I	I	I-DIMENSION, 1ST SUBSCRIPT LIMIT (0 IF NONE)
SUB2	O	I	I	J-DIMENSION, 2ND SUBSCRIPT LIMIT (0 IF NONE)
EQUAT	O	I	I	EQUAL/AT FLAG 0=EQUAL SIGN ENCOUNTERED 1=AT SIGN ENCOUNTERED
BPTR	O	I	I	INDEX TO NEXT FIELD IN BUFF
STAT	O	I	I	STATUS FLAG FOR MDEDIT PROCESSING 0=NORMAL RETURN 5=SYNTAX ERROR

EXTERNAL REFERENCES
NONE

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON

NONE

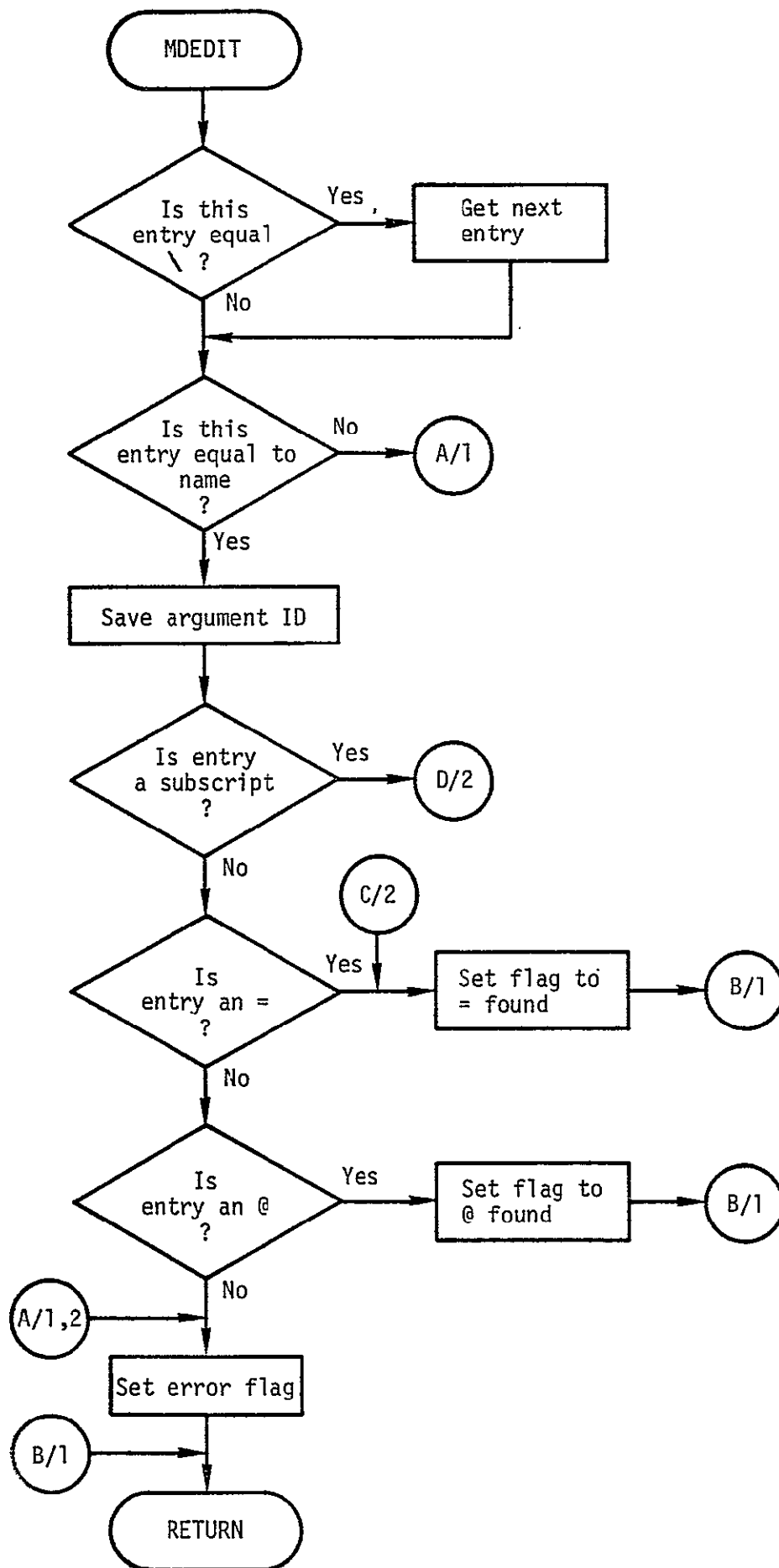
COMMON /MDCODE/

AT	I
BCKSLH	I
EQUALS	I
NAME	I
SUBS	I

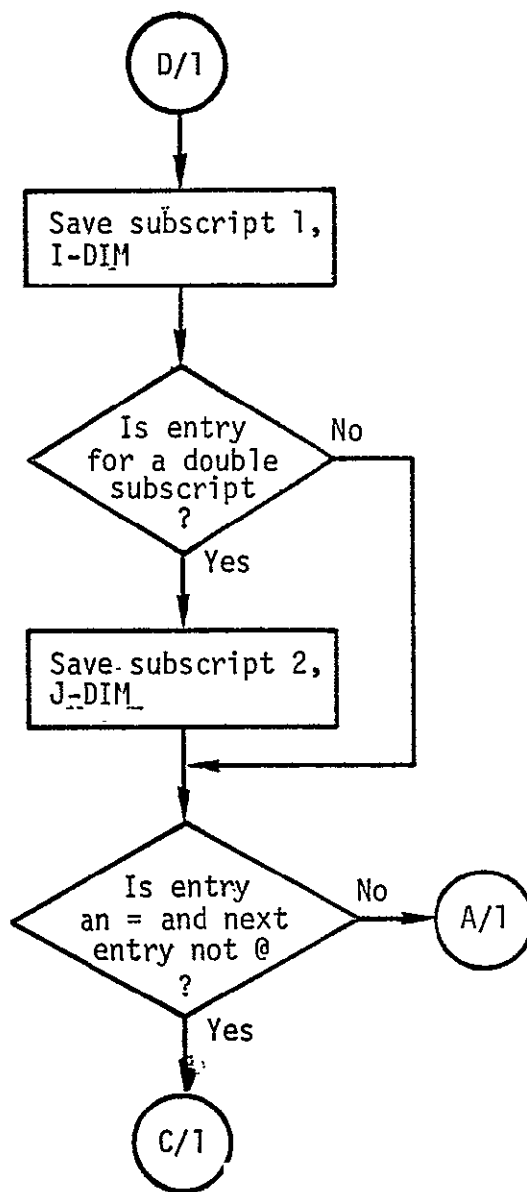
LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDEDIT Flow Diagram
7.7-3



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDSPEC - Control Table Editor

The purpose of MDSPEC is to return the argument specifications of a particular argument of a control table.

Method

Input: The input to MDSPEC is the control table and the argument number passed through the calling sequence.

Processing: MDSPEC calculates from the argument number the index into the control table, locates, and stores data for output.

Output: The output from MDSPEC consists of the following control table information:

Argument identifier (name)

I-dimension

J-dimension

Type of variable

Constant flag

I/O flag

Completion flag

Print flag

@ indicator flag

USAGE
 ENTRY MDSPEC
 CALL MDSPEC (CTAB, ARGNUM, ARGID, IDIM, JDIM, TYPE, CONST, IOFLG,
 COMPL, PRNTFLG, EQUAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
CTAB	I	I	I	CONTROL TABLE
ARGNUM	I	I	I	ARGUMENT NUMBER
ARGID	O	I	I	ARGUMENT IDENTIFIER (NAME)
IDIM	O	I	I	I-DIMENSION, 1ST SUBSCRIPT LIMIT
JDIM	O	I	I	J-DIMENSION, 2ND SUBSCRIPT LIMIT
TYPE	O	I	I	TYPE OF VARIABLE
CONST	O	I	I	CONSTANT FLAG
IOFLG	O	I	I	I/O FLAG
COMPL	O	I	I	COMPLETION FLAG
PRNTFLG	O	I	I	PRINT FLAG
EQUAT	O	I	I	"/" INDICATOR

EXTERNAL REFERENCES
 NONE

DIAGNOSTICS
 NONE

EXTERNAL STORAGE
 NONE

BLANK COMMON
 NONE

LOCAL COMMON
 NONE

REPRODUCIBILITY OF THE
 ORIGINAL PAGE IS POOR

MDALCT - Utility

MDALCT performs the function of the utility processor ALOCAT, i.e., allocate an array in the storage monitor table.

Method

Input: The calling arguments to MDALCT are not set up when MDALCT is entered; however, they are in the working buffer and MDALOC will be called within MDALCT to set up the arguments. The calling arguments are the Ith and Jth dimension, type and name of the array.

Processing: MDALOC must be called to set up the calling arguments; however, the name of the array was specified with an @ sign denoting to MDALOC to allocate a SMT entry for the array name. This might cause an error; therefore, MDALCT must modify the name, which is the fourth argument, to a = sign denoting immediate data. MDALOC is then called to set up the arguments.

The input type representation (R, I, H or F) is changed to the internal integer form. If the array name with any type exist in the SMT, it is deleted and a message is printed. In any case, a new SMT entry is allocated for the array name, type, and size. The data area for that SMT entry is set to zero.

Output: The output from MDALCT is an entry in the SMT for the array with the given dimension, name and type with the data area cleared. A status flag is also output.

SAGE

ENTRY MDALCT

CALL MDALCT (STATUS)

ARGMT I/O TYPE DIM

DEFINITION

STATUS 0 1

STATUS FLAG

> 0 NORMAL RETURN

> 2 COULD NOT FIND ARRAY NAME

> 4 MEMORY NOT AVAILABLE

> 5 COULD NOT DELETE PREVIOUS ARRAY
WITH THE SAME NAME.

EXTERNAL REFERENCES

MDLKUP

MDALOC

MDFIND

MDELETE

MDENTR

DIAGNOSTICS

.....(TYPE) DELETED.

THE ARRAY NAME ALREADY EXISTED. THE SMT ENTRY FOR THE
OLD ARRAY NAME AND TYPE HAS BEEN DELETED.

EXTERNAL STORAGE :

NONE

BLANK COMMON

VARB I/O

ARGADD 0

COMMON /MDBUFF/

VARB I/O

BDATA I

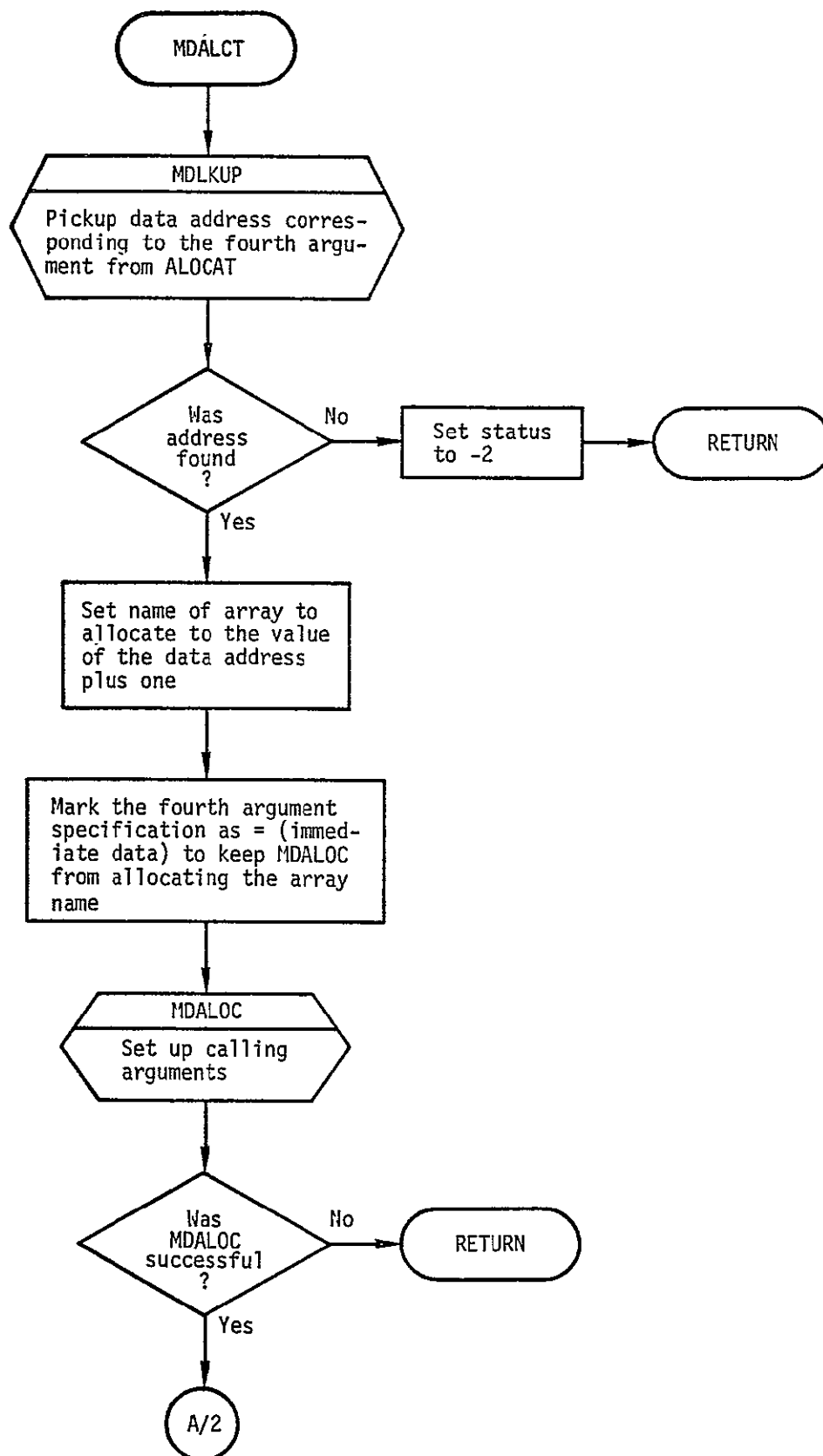
DSIZE I

WBUF I

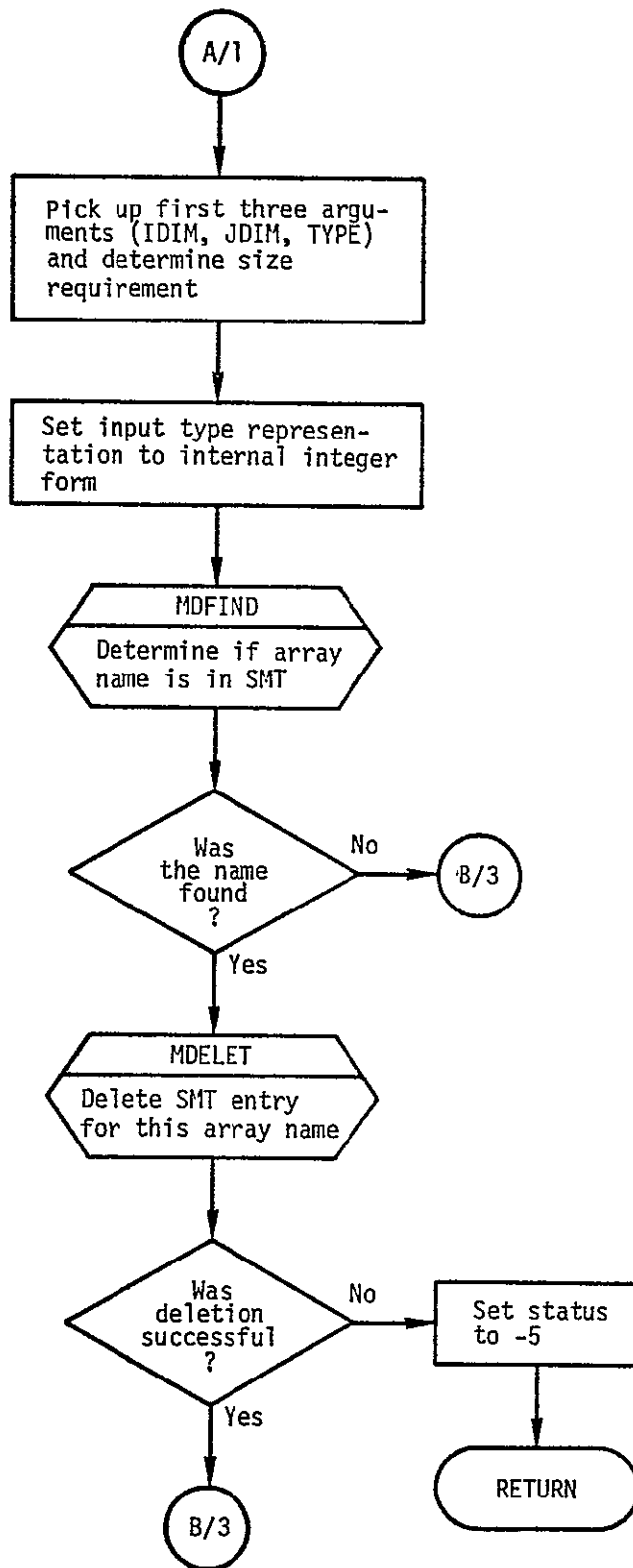
LOCAL COMMON

NONE

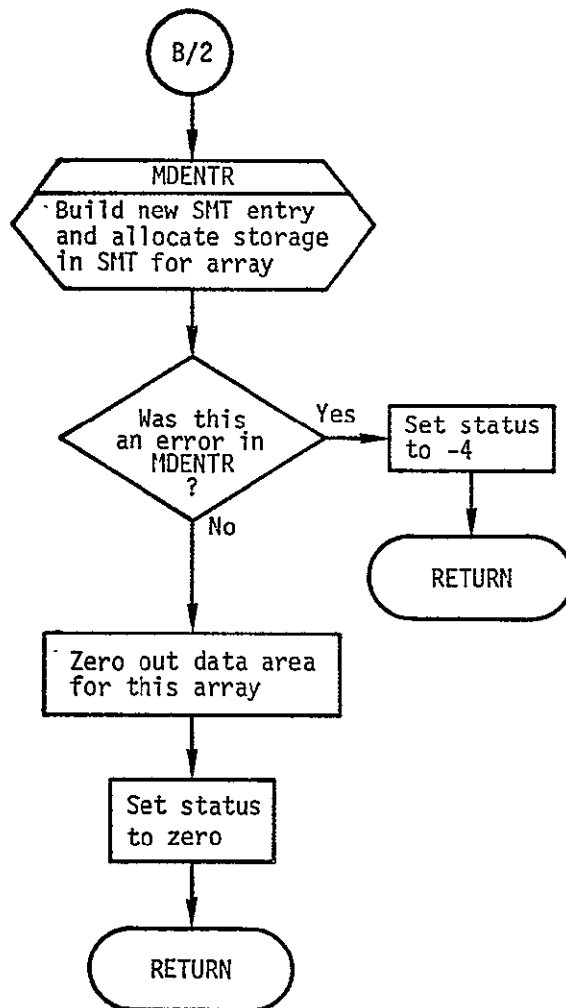
REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDALCT Flow Diagram



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDALCT Flow Diagram (Continued)

MDCTPK - Utility

MDCTPK is the routine which packs a control table after it has been split into two parts, the argument specifications and the data (see also, MDSPLT).

Method

Input: All input to this routine is contained in the common block MDBUFF and consists of: the length of the working buffer, an index to the data portion of the control table, the amount (in words) used and the control table (in the working buffer).

Processing: All arguments with data in the buffer are scanned beginning with the one which appears "highest" in the buffer. Once an argument's data has been processed (i.e., moved to the area immediately below the argument specifications) all subsequent appearances of this argument's data are ignored. If an argument's data is not complete it is ignored also. Upon completion of the pack, all data lies immediately below the argument specifications with an argument's data appearing at most once.

Output: The output is also contained in common block MDBUFF and consists of: the packed control table (still in the working buffer) and an index to the first word of the argument data.

USAGE

ENTRY MDCTPK
CALL MDCTPK

EXTERNAL REFERENCES
NONE

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON

NONE

COMMON / MDBUFF /

.VARB	I/O
-------	-----

MDLEN	I
-------	---

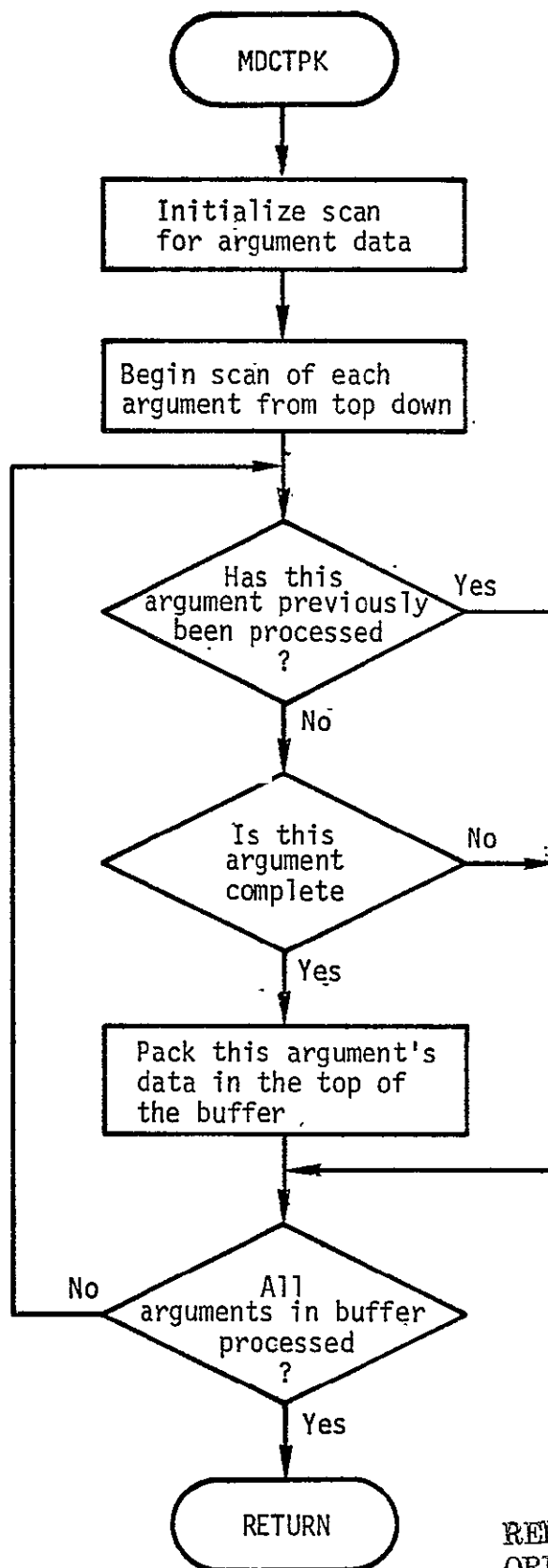
BDATA	I
-------	---

DSIZE	I
-------	---

WBUF	I
------	---

LOCAL COMMON

NONE



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDCTPK Functional Flow Diagram

MDGETC - Utility

MDGETC is used to input data from files by file name. Each input results in a single record being placed in a buffer supplied by the calling routine.

Method

Input: The calling routine supplies MDGETC with the name and version of the file, an option flag specifying the specific action to be taken and a buffer to receive the data from a single logical record.

Processing: There are four options to MDGETC: open the file, input a single record and close the file; open the file and input the first record; input subsequent records to an opened file; and close an opened file. On an open option MDGETC sets up the appropriate control block parameters. For options using previously opened files the name and version are checked for match to verify the validity of the current control blocks. Except for the close option a record is input each time MDGETC is called.

Output: A status flag is returned indicating successful execution, read error returned from the system services functions, opening of a non-existent file was attempted, the logical record was truncated to the buffer length, or improper input to MDGETC.

USAGE

ENTRY MDGETC

CALL MDGETC (FILNAM, VER, OPTION, LENGTH, BUFFER, STATUS)

ARGMT I/O TYPE DIM

DEFINITION

FILNAM	I	H	1	SIX CHARACTER FIELD DATA NAME
VER	I	H	1	TWO CHARACTER FIELD DATA VERSION
OPTION	I	I	1	INPUT OPTION FLAG
				=1, OPEN FILNAM.VER, INPUT RECORD INTO BUFFER AND CLOSE FILE
				=2, OPEN FILNAM.VER AND INPUT RECORD INTO BUFFER
				=3, INPUT RECORD FROM PREVIOUSLY OPENED FILNAM.VER
				=4, CLOSE PREVIOUSLY OPENED FILNAM.VER
LENGTH	I	I	1	LENGTH (IN WORDS) OF RECORD TO BE INPUT INTO BUFFER
BUFFER	O	F	LENGTH	CONTENTS OF INPUT RECORD
STATUS	O	I	1	COMPLETION STATUS
				= 0, NORMAL COMPLETION
				=-1, FILNAM.VER NOT FOUND
				=-2, RECORD TRUNCATED TO LENGTH WORDS ON INPUT
				=-3, READ ERROR
				=-4, INVALID OPTION
				=-5, FILNAM.VER OF OPTION 3 OR 4 DOES NOT MATCH THAT OF PREVIOUS CALL

EXTERNAL REFERENCES

ECLOS\$ TO CLOSE FILES
 ELRSR\$ TO READ LOGICAL RECORDS
 EOPENS\$ TO OPEN FILES
 FWKBK\$ TO GENERATE WALK BACKS AND TERMINATE EXECUTION
 MDCONV TO CONVERT FROM FIELD DATA TO ASCII

RESTRICTIONS

ALL INPUT FROM A FILE DURING ONE OPEN MUST BE ACCOMPLISHED VIA MDGETC.
 ONLY ONE OPEN FILE AT A TIME IS SUPPORTED WITHIN MDGETC

DIAGNOSTICS

NONE

EXTERNAL STORAGE

THE REQUESTED I/O ACTIVITIES ARE ACCOMPLISHED ON THE DESIGNATED FILE

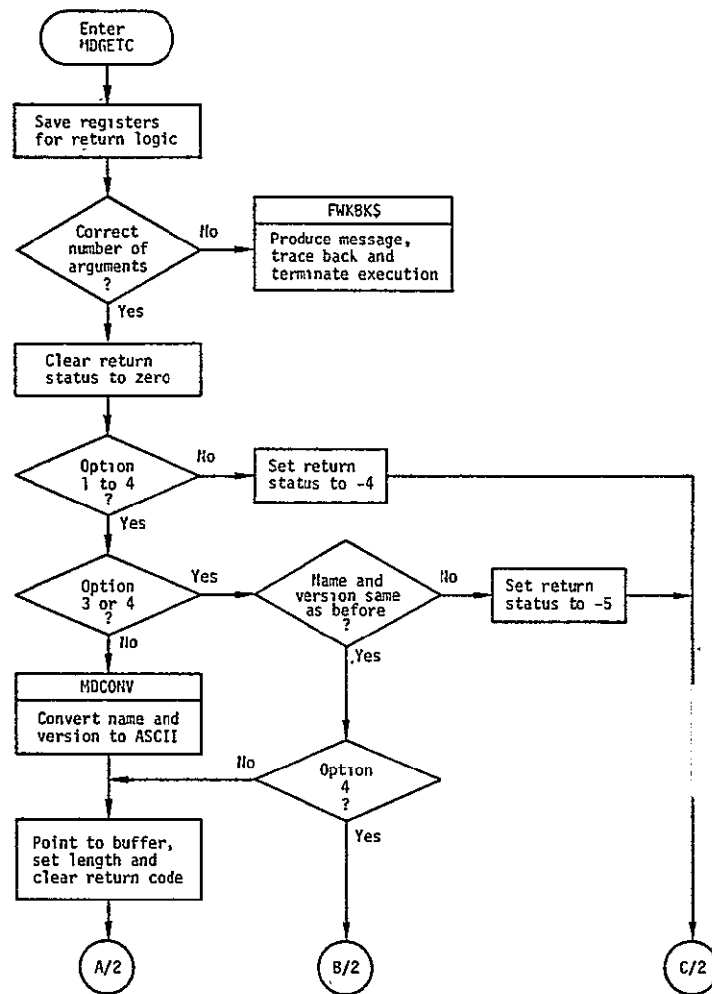
BLANK COMMON

NONE

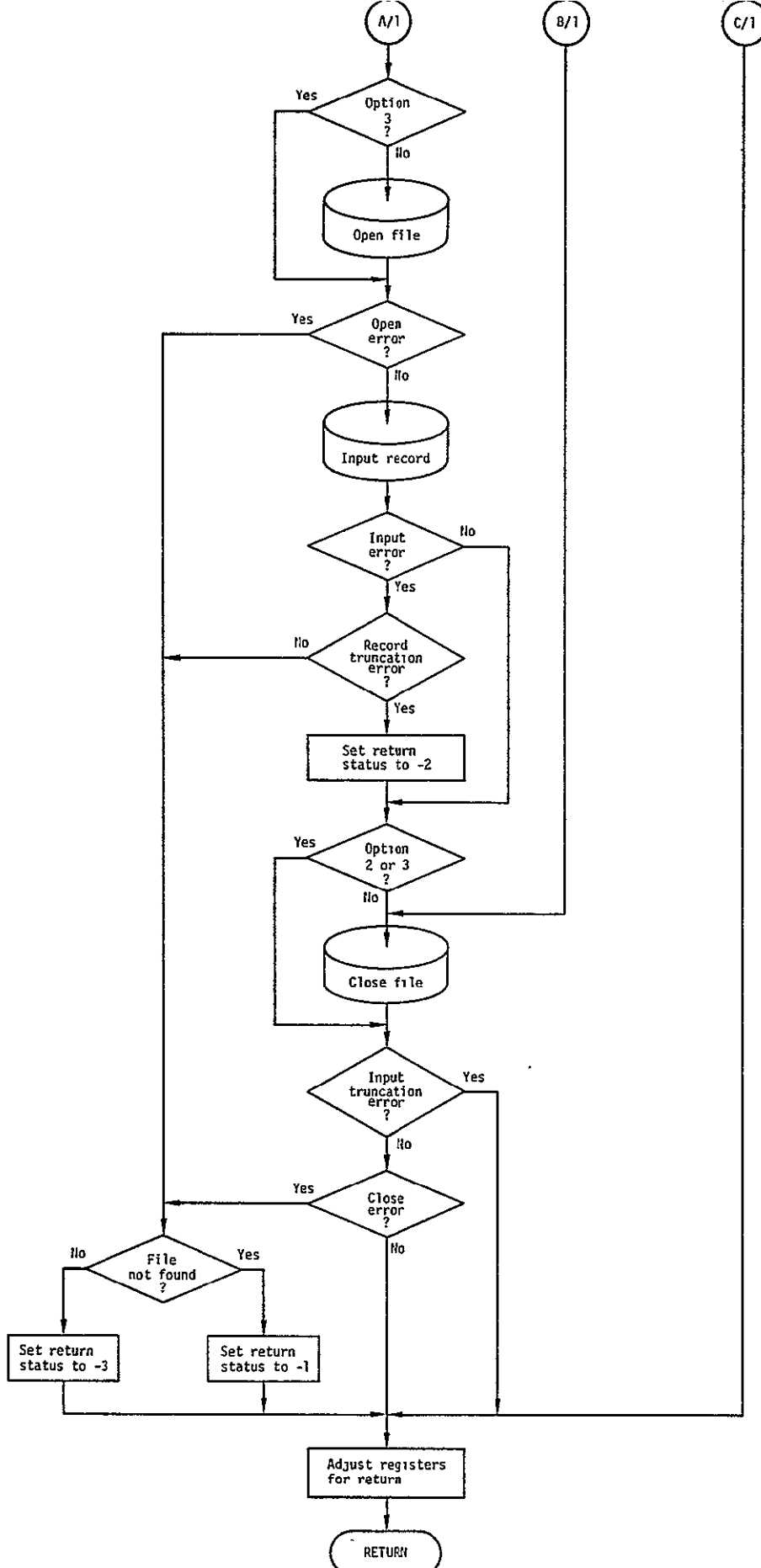
LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
 ORIGINAL PAGE IS POOR



HDGETC - File Input Utility Routine Functional Flow



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDLIST - Utility

MDLIST is called by the submonitor (MDSMON) to list the various data elements (command tables, control tables and data arrays).

Method

Input: All input is through the calling sequence and consists of: a buffer containing the data to be listed, the type designator of the data element, the size (in words) of the data and the data's alphanumeric identifier.

Processing: If the type designator indicates a data array, the type is further broken down to determine if the array should be listed in octal, real, integer or Hollerith format. After performing the indicated list, control is returned to the submonitor.

If the type indicated a control table, all argument specifications are listed. Any complete argument also has its current values listed. Control is then returned to the submonitor.

If a command table is to be listed, a heading is printed out and the routine MDCMTL is called to perform the list. Once again, control is returned to the submonitor.

Output: There is no output from MDLIST other than the requested listing.

USAGE

ENTRY MDLIST

CALL MDLIST (BUFF,TYPE,SIZE,NAME)

ARGMT	I/O	TYPE	DIM	DEFINITION
BUFF	I	I	SIZE	BUFFER CONTAINING DATA ELEMENT TO BE LISTED
TYPE	I	I	I	TYPE DESIGNATOR OF DATA ELEMENT
SIZE	I	I	I	LENGTH IN WORDS OF DATA ELEMENT
NAME	I	I	I	ALPHANUMERIC DESIGNATOR OF DATA ELEMENT

EXTERNAL REFERENCES

MDALST
MDCMTL
MDLSTH
MDLSTI
MDLSTO
MDLSTR
MDSPEC

DIAGNOSTICS

NONE

EXTERNAL STORAGE

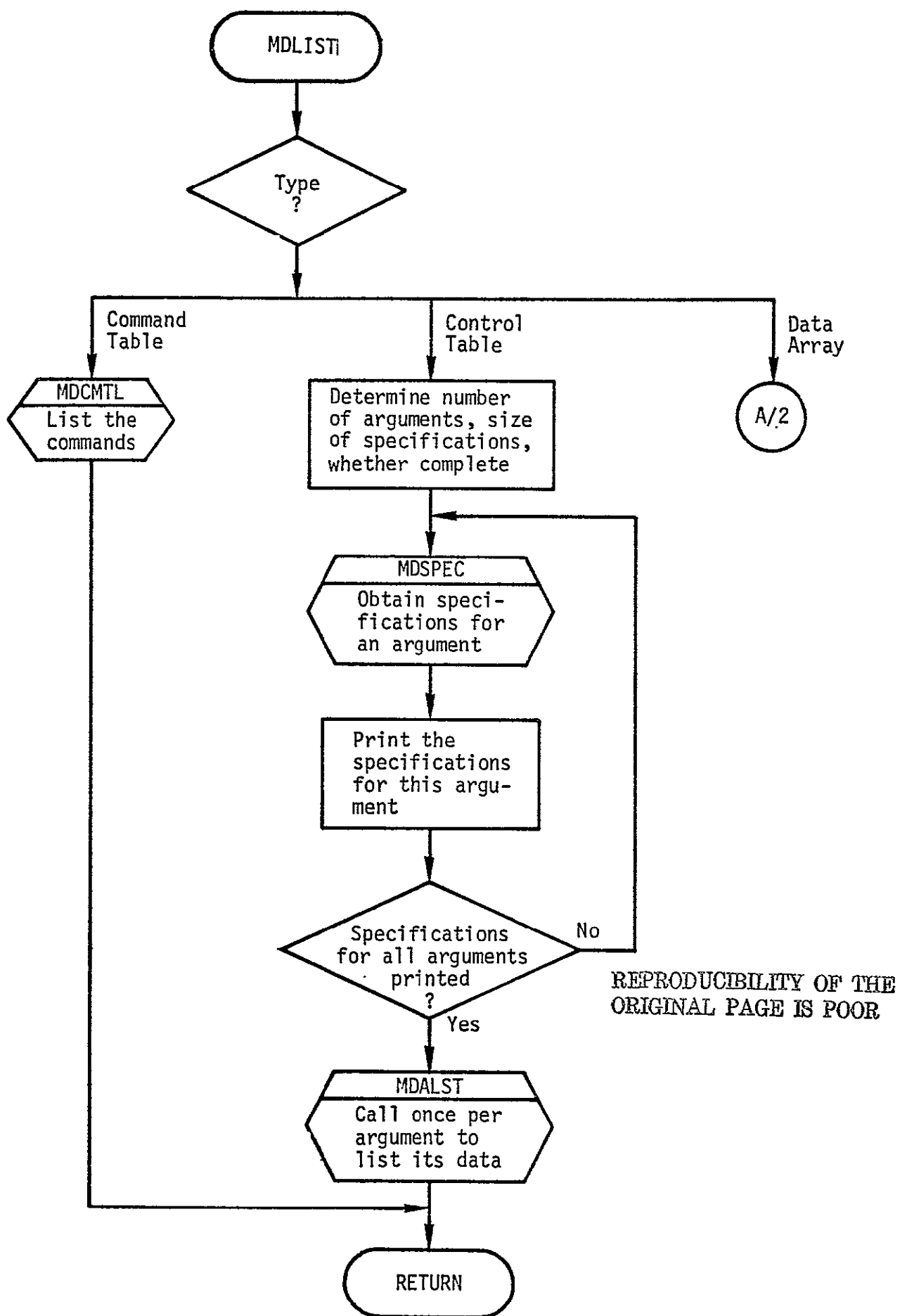
NONE

BLANK COMMON

NONE

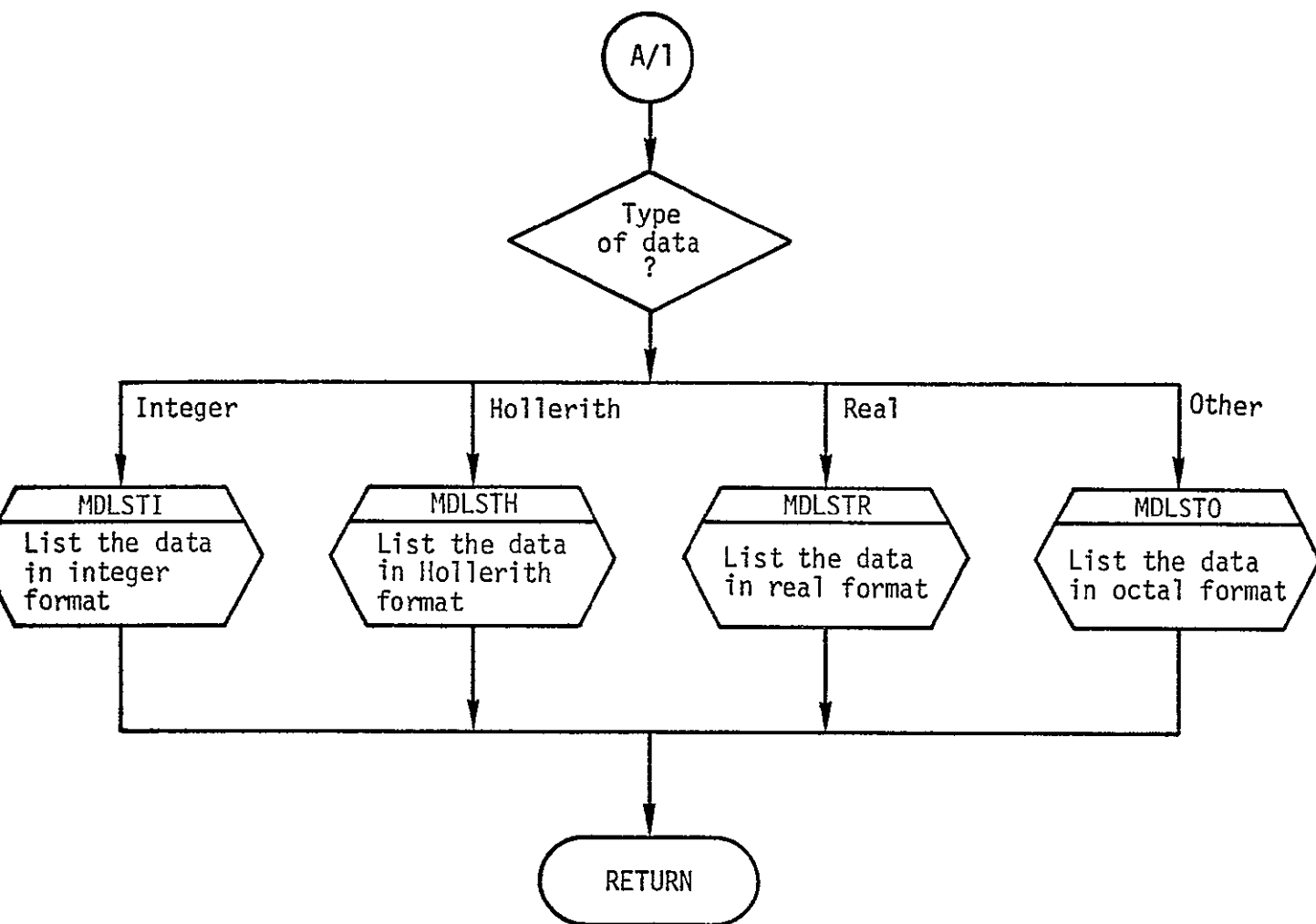
LOCAL COMMON

NONE



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDLIST Flow Diagram



MDLIST Flow Diagram (Continued)

MDLKUP - Utility

MDLKUP is the routine used to locate the data portion of an argument in a control table.

Method

Input: The input to MDLKUP is through the calling sequence and consists of: the argument label of the data to be found, the buffer containing the control table, an index to the beginning word of the data portion of the control table and the length (in words) of the control table.

Processing: Each data sets identifier is compared to the input argument label. When a match is found, the search is terminated and the current location is returned. If no match was found, an error indication is returned to the calling routine.

Output: All output is through the calling sequence and consists of the data's location in the control table and a status flag. If the status flag is non-zero the location indicator has no meaning.

USAGE

ENTRY MDLKUP

CALL MDLKUP (ANUM,WBUF,BDATA,LDATA,DATADD,STATUS)

ARGMT	I/O	TYPE	DIM	DEFINITION
ANUM	I	I	1	ARGUMENT LABEL TO BE LOCATED
WBUF	I	I	VARB	BUFFER CONTAINING THE CONTROL TABLE
BDATA	I	I	1	BEGINNING OF THE DATA PORTION OF THE BUFFER
LDATA	I	I	1	LENGTH(IN WORDS) OF THE BUFFER
DATADD	O	I	1	LOCATION OF THE ARGUMENT DATA
STATUS	O	I	1	STATUS FLAG; =0, RETURN OK =-1, ARGUMENT DATA NOT FOUND

EXTERNAL REFERENCES

NONE

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

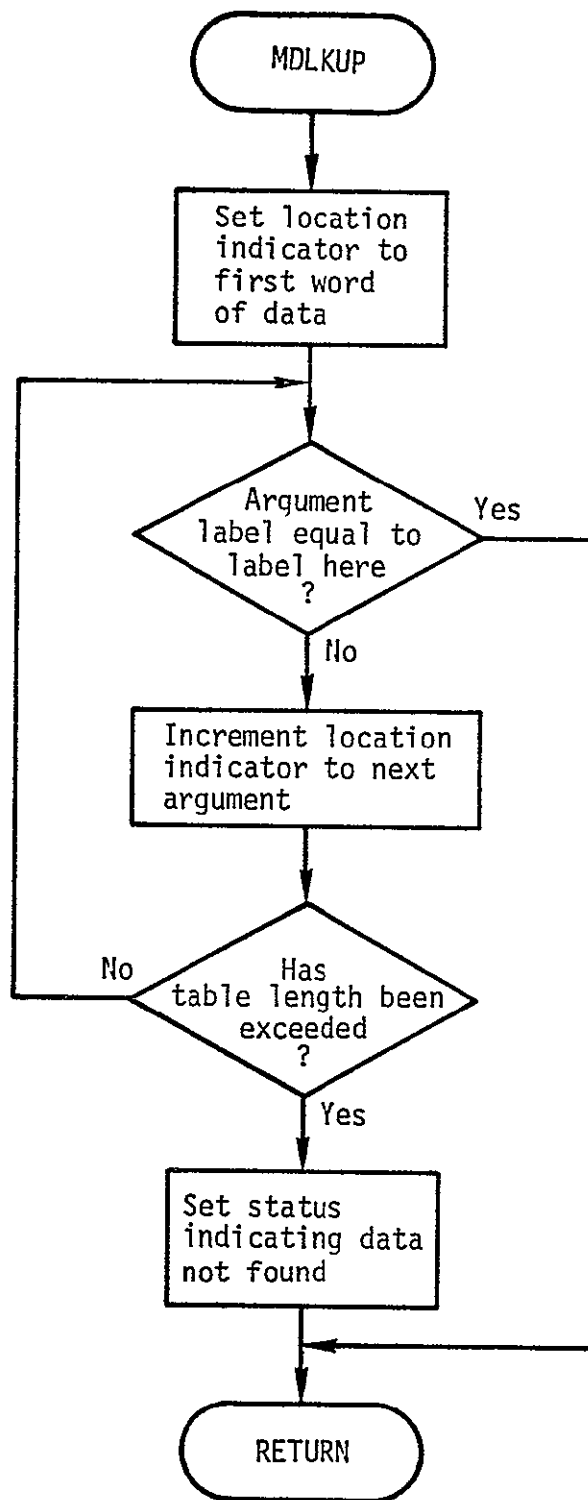
BLANK COMMON

NONE

LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDLKUP Flow Diagram

MDLSTH - Utility Support Routine

This routine will list arbitrary amounts of Hollerith data.

Method

Input: All input is through the calling sequence and consists of: the argument identifier of data of the data being listed, an array containing the data and the amount of data (in words) to be listed.

Processing: If possible, all data is printed on one line; if not, then there are 10 words per line. The argument identifier appears on the first line only.

Output: There is no output from this routine other than the listing of data for the user.

USAGE

ENTRY MDLSTH

CALL MDLSTH(NAME,ARRAY,LEN)

ARGMT	I/O	TYPE	DIM	DEFINITION
NAME	I	I	I	ARGUMENT NAME OF THE DATA TO BE LISTED
ARRAY	I	I	VARB	ARRAY CONTAINING THE DATA TO BE LISTED
LEN	I	I	I	AMOUNT (IN WORDS) TO BE LISTED

EXTERNAL REFERENCES
NONE

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE

MDLSTI - Utility Support Routine

This routine will list arbitrary amounts of integer data.

Method

Input: All input is through the calling sequence and consists of: the argument identifier of data of the data being listed, an array containing the data and the amount of data (in words) to be listed.

Processing: If possible, all data is printed on one line; if not, then there are 6 words per line. The argument identifier appears on the first line only.

Output: There is no output from this routine other than the listing of data for the user.

USAGE

ENTRY MDLSTI

CALL MDLSTI(NAME,ARRAY,LEN)

ARGMT	I/O	TYPE	DIM	DEFINITION
-------	-----	------	-----	------------

NAME	I	I	I	ARGUMENT NAME OF THE DATA TO BE LISTED
------	---	---	---	--

ARRAY	I	I	VARB	ARRAY CONTAINING THE DATA TO BE LISTED
-------	---	---	------	--

LEN	I	I	I	AMOUNT(IN WORDS) TO BE LISTED
-----	---	---	---	-------------------------------

EXTERNAL REFERENCES

NONE

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

NONE

LOCAL COMMON

NONE

MDLSTO - Utility Support Routine

This routine will list arbitrary amounts of octal data.

Method

Input: All input is through the calling sequence and consists of: the argument identifier of data of the data being listed, an array containing the data and the amount of data (in words) to be listed.

Processing: If possible, all data is printed on one line; if not, then there are 4 words per line. The argument identifier appears on the first line only.

Output: There is no output from this routine other than the listing of data for the user.

USAGE

ENTRY MDLSTO

CALL MDLSTO(NAME,ARRAY,LEN)

ARGMT	I/O	TYPE	DIM	DEFINITION
NAME	I	I	I	ARGUMENT NAME OF THE DATA TO BE LISTED
ARRAY	I	FREE	VARB	ARRAY CONTAINING THE DATA TO BE LISTED
LEN	I	I	I	AMOUNT(IN WORDS) TO BE LISTED

EXTERNAL REFERENCES

NONE

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

NONE

LOCAL COMMON

NONE

MDLSTR - Utility Support Routine

This routine will list arbitrary amounts of real data.

Method

Input: All input is through the calling sequence and consists of: the argument identifier of data of the data being listed, an array containing the data and the amount of data (in words) to be listed.

Processing: If possible, all data is printed on one line; if not, then there are 4 words per line. The argument identifier appears on the first line only.

Output: There is no output from this routine other than the listing of data for the user.

USAGE

ENTRY MDLSTR

CALL MDLSTR(NAME,ARRAY,LEN)

ARGMT	I/O	TYPE	DIM	DEFINITION
-------	-----	------	-----	------------

NAME	I	I	I	ARGUMENT NAME OF THE DATA TO BE LISTED
------	---	---	---	--

ARRAY	I	R	VARB	ARRAY CONTAINING THE DATA TO BE LISTED
-------	---	---	------	--

LEN	I	I	I	AMOUNT (IN WORDS) TO BE LISTED
-----	---	---	---	--------------------------------

EXTERNAL REFERENCES

NONE

DIAGNOSTICS

NONE

EXTERNAL STORAGE

NONE

BLANK COMMON

NONE

LOCAL COMMON

MDPUTC - Utility

MDPUTC is used to output data to files by file name. Each output generates a single record from a buffer supplied by the calling routine.

Method

Input: The calling routine supplies MDPUTC with the name and version of the file, an option flag specifying the specific action to be taken and a buffer of data to be output as a single logical record.

Processing: There are four options to MDPUTC: open the file, output a single record and close the file; open the file and output the first record; output subsequent records to an opened file; and close an opened file. On an open option MDPUTC sets up the appropriate control block parameters. For options using previously opened files the name and version are checked for match to verify the validity of the current control blocks. Except for the close option a record is created in the file each time MDPUTC is called.

Output: A status flag is returned indicating successful execution, write error returned from the system services function or improper input to MDPUTC.

USAGE

ENTRY MDPUTC

CALL MDPUTC (FILNAM, VER, OPTION, LENGTH, BUFFER, STATUS)

ARGMT	I/O	TYPE	DIM	DEFINITION
FILNAM	I	H	1	SIX CHARACTER FIELD DATA NAME
VER	I	H	1	TWO CHARACTER FIELD DATA VERSION
OPTION	I	I	1	OUTPUT OPTION FLAG =1, OPEN FILNAM.VER, OUTPUT BUFFER AS A SINGLE RECORD AND CLOSE FILE =2, OPEN FILNAM.VER AND OUTPUT BUFFER =3, OUTPUT BUFFER INTO PREVIOUSLY OPENED FILNAM.VER =4, CLOSE PREVIOUSLY OPENED FILNAM.VER
LENGTH	I	I	1	LENGTH (IN WORDS) OF RECORD TO BE OUTPUT FROM BUFFER
BUFFER	I	F	LENGTH	ARRAY CONTAINING LENGTH WORDS TO BE OUTPUT AS A SINGLE LOGICAL RECORD INTO FILE FILNAM.VER
STATUS	O	I	1	COMPLETION STATUS = 0, NORMAL COMPLETION =-3, WRITE ERROR =-4, INVALID OPTION =-5, FILNAM.VER OF OPTION 3 OR 4 DOES NOT MATCH THAT OF PREVIOUS CALL

EXTERNAL REFERENCES

ECLOSE\$ TO CLOSE FILES
ELRSW\$ TO WRITE LOGICAL RECORDS
EOPENS\$ TO OPEN FILES
FWKBK\$ TO GENERATE WALK BACKS AND TERMINATE EXECUTION
MDCONV TO CONVERT FROM FIELD DATA TO ASCII

RESTRICTIONS

ALL OUTPUT TO A FILE DURING ONE OPEN MUST BE ACCOMPLISHED
VIA MDPUTC.
ONLY ONE OPEN FILE AT A TIME IS SUPPORTED WITHIN MDPUTC.

DIAGNOSTICS

NONE

EXTERNAL STORAGE

THE REQUESTED I/O ACTIVITIES ARE ACCOMPLISHED ON THE
DESIGNATED FILE

BLANK COMMON

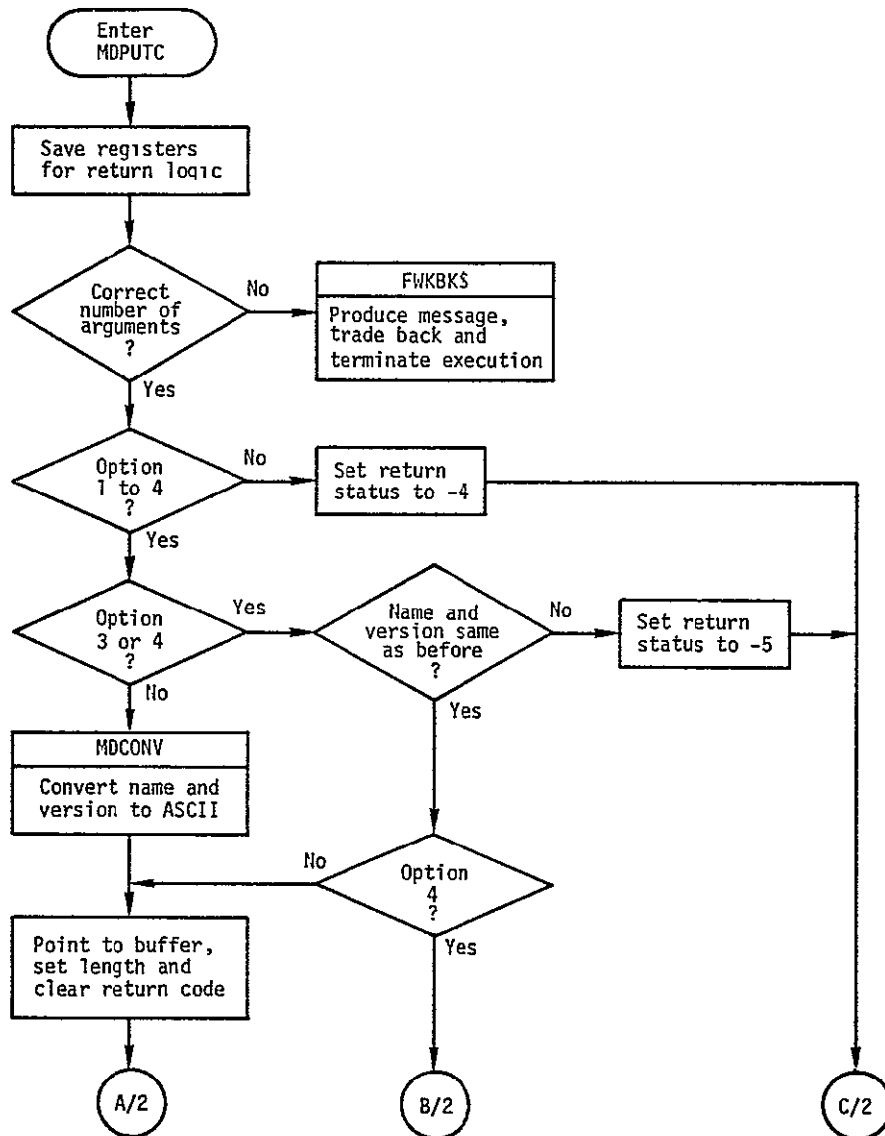
VARB I/O

OCB 0
UCB 0

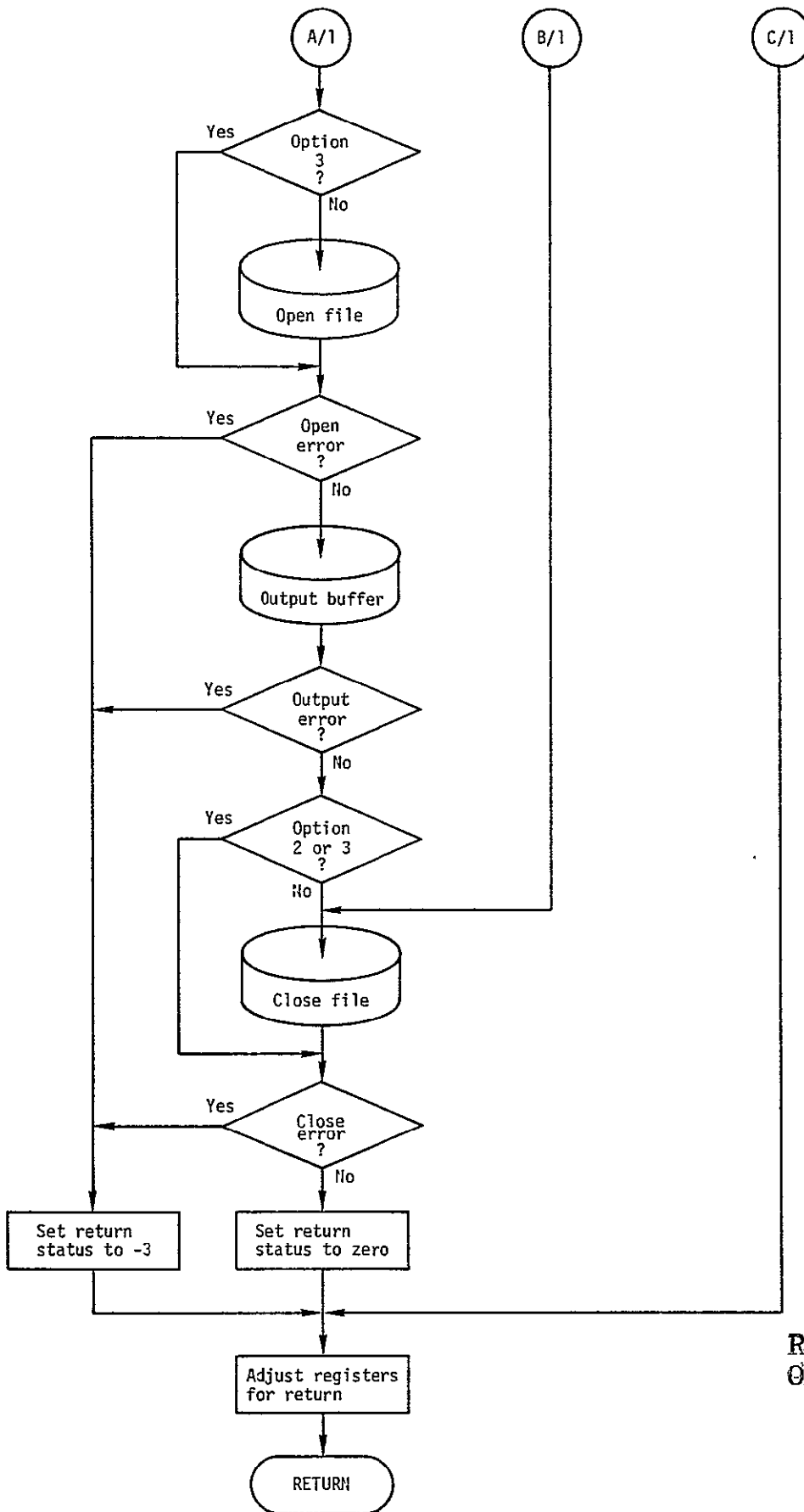
LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDPUTC - File Output Utility Routine Functional Flow



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDQUIT - Utility Routine

This routine performs the QUIT directive and determines if the user desires to save a SMT.

Method

Input: There is no input to this routine.

Processing: Upon entry, the user is prompted to determine if he desires to save the SMT. If he does not, control is returned to MDSMON where a STOP statement is executed to terminate the session. If he does desire to save the SMT, all entries prefixed by \$ and % are deleted, these entries residing in the IMS data base. The entries, &CMDTB and &CONTB, are deleted also. Control is now passed to MDSMTW for the writing of the SMT to a file. Upon return from MDSMTW, control is returned to MDSMON where execution is terminated.

Output: This routine has no output.

USAGE

ENTRY MDQUIT
CALL MDQUIT

EXTERNAL REFERENCES

MDELET
MDPACK
MDPRMT
MDSMTW

DIAGNOSTICS

I/O ERROR WHILE PROMPTING
A SYNTAX ERROR HAS BEEN ENCOUNTERED WHILE DETERMINING
IF A FILE IS TO BE SAVED.

EXTERNAL STORAGE

NONE

BLANK COMMON

VARB I/O

DBSTRT I
NTRY I

COMMON / MDCODE /

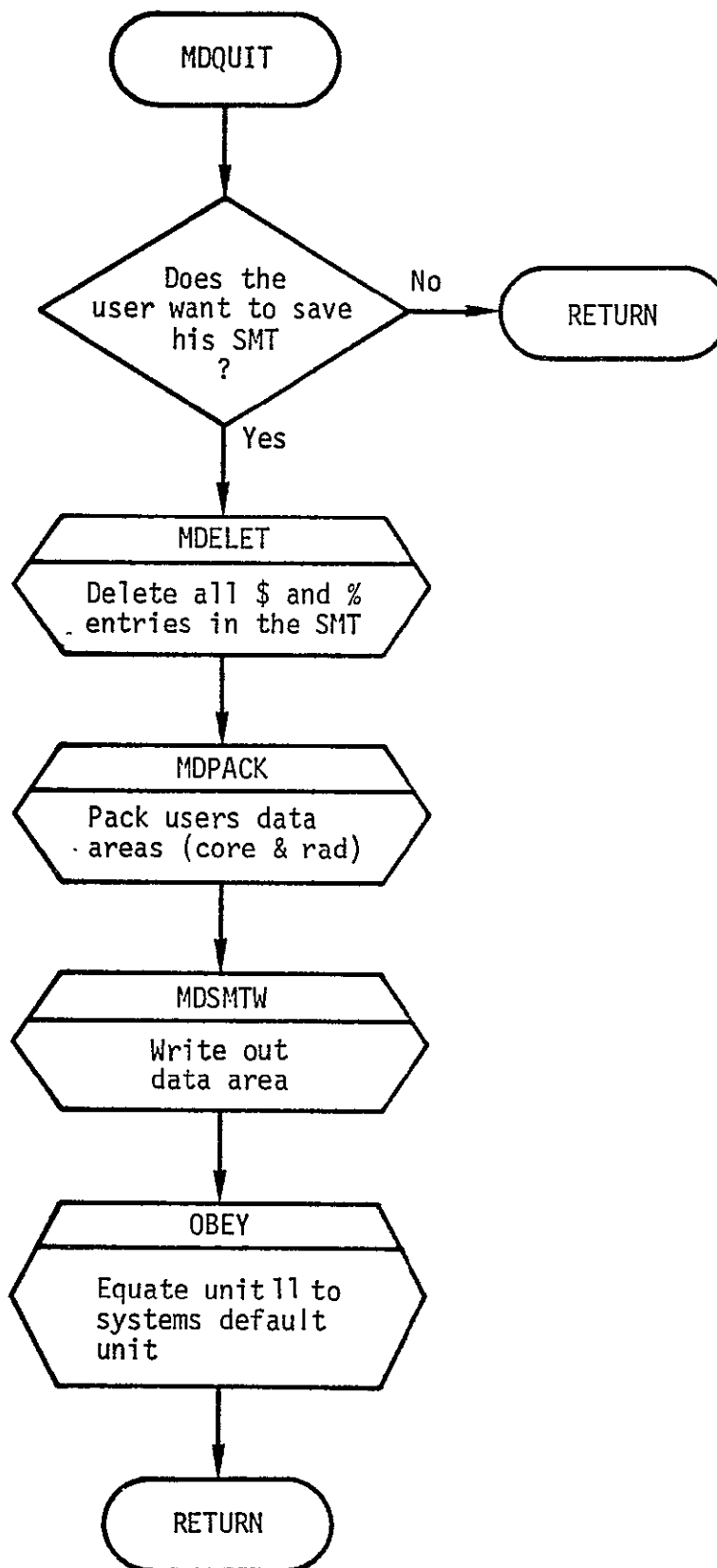
VARB I/O

NAME I

LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDQUIT Functional Flow Diagram

MDSMTW - Utility Support Routine

MDSMTW is a routine with two entry points. One causes the current SMT to be written to a file. The other causes a user specified file to be read into the SMT.

Method

Input: A buffer is passed through the calling sequence containing the file name to be read from or written to. If no name is present the first word of the buffer is -1.

Processing: If the user desires to save a file, either through the SAVE directive or the QUIT directive, the MDSMTW entry point is called. A check is made for the presence of a file name and, if not present, the user is asked to provide one. The user must input a file name. The first record of the file, containing information concerning where the data starts, how much data is present and the maximum size allowed, is written out. The data is then packed and written as the second record. The file is now closed. While packing the data, there is a possibility of destroying part of the SMT in core. Therefore, any portion that was destroyed is restored to the condition it was in upon entry. Control is now returned to the calling program.

If the user desires to read a previously saved file, either at initialization time or with the RECALL directive, the MDSMTR entry point is called. Once again, a check is made for the presence of a file name and, if not present, the user is asked to provide one. If the user is specifying a file saved under another access code, the access file (MDACCD) is read into the working buffer to obtain the version the file was saved with. The first record of the desired file is read into blank common. The record contains information regarding the attributes of the data. If the file will not fit in the current configuration the user is informed and the reading process terminated. Otherwise, the second record is read into blank common, moved to the bottom and each entry's address field is adjusted. The file is now closed and control is returned to the calling routine.

Output: There is no output from either entry point (other than the file read/written).

USAGE

ENTRY MDSMTW
CALL MDSMTW(INPUT)

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

ARGMT I/O TYPE DIM

DEFINITION

INPUT I I VARB BUFFER CONTAINING THE DIRECTIVE

ENTRY MDSMTR
CALL MDSMTR(INPUT)

INPUT I I VARB BUFFER CONTAINING THE DIRECTIVE

EXTERNAL REFERENCES

MDGETC

MDPRMT

MDPUTC

MDSPLT

DIAGNOSTICS

***COULD NOT OPEN AND READ ACCESS FILE**

THE USER HAS SPECIFIED A SECONDARY ACCESS CODE OTHER THAN HIS OWN. IN ATTEMPTING TO READ THE FILE OF ACCESS CODES AN ERROR OCCURED.

*** COULD NOT OPEN AND READ , STATUS = ...

THE USER SPECIFIED FILE COULD NOT BE OPENED AND READ INTO MEMORY.

*** COULD NOT OPEN AND WRITE TO , STATUS = ...

THE USER HAS ATTEMPTED TO SAVE AN EXISTING FILE. AN ERROR OCCURED WHEN THE WRITE WAS ATTEMPTED.

*** DATA BASE NOT SAVED/RECOVERED

ANY I/O ERROR IN READING OR WRITING A FILE HAS THIS MESSAGE APPENDED TO IT.

*** ERROR IN CLOSINGSTATUS=*** INTEGRITY OF DATA BASE IS QUESTIONABLE

WHILE ATTEMPTING TO CLOSE THE USER SPECIFIED FILE AN ERROR OCCURED.(READING ONLY)

*** I/O ERROR IN WRITING OR CLOSING STATUS=... , INTEGRITY OF FILE IS QUESTIONABLE

SAME AS PRECEDING EXCEPT WRITE ONLY

*** LENGTH OF SAVED DATA BASE (....) EXCEEDS MAXIMUM (....) OF CURRENT CONFIGURATION.

AN ATTEMPT HAS BEEN MADE TO RECOVER A DATA AREA THAT IS TOO LARGE FOR CURRENT SYSTEM SIZE

*** READ ERROR IN READING DATA PORTION OF , STATUS=...

AN ERROR HAS OCCURED WHEN READING THE SECOND RECORD OF A TWO RECORD FILE.

*** READ ERROR WHILE READING RESPONSE

AN ERROR HAS OCCURED WHILE PROMPTING FOR THE USER'S FILE NAME

***-SECONDARY CODE NOT FOUND

THE USER HAS ATTEMPTED TO READ A FILE WITH AN INVALID ACCESS CODE

***SYNTAX ERROR--FILE NAME ONLY IS ALLOWED

THE USER HAS ATTEMPTED TO SAVE A FILE TO WHICH HE HAS APPENDED AN ACCESS CODE

***SYNTAX ERROR--TRY AGAIN--**

THE USER HAS MADE A SYNTAX ERROR WHEN TRYING TO RECALL A FILE.

EXTERNAL STORAGE
MDACCD
VARIOUS OTHER FILES

FILE CONTAINING ACCESS CODES
USER SPECIFIES THE NAME OF THE FILE
IN EXTERNAL STORAGE CONTAINING HIS
DATA

BLANK COMMON
VARB 1/0

DBADDR	I
DBMAX	I
DBSTRT	I
NTRY	I
ACCCDE	I
BDGNUM	I
NENTR	I
VERS	I

COMMON / MDCODE /

VARB 1/0

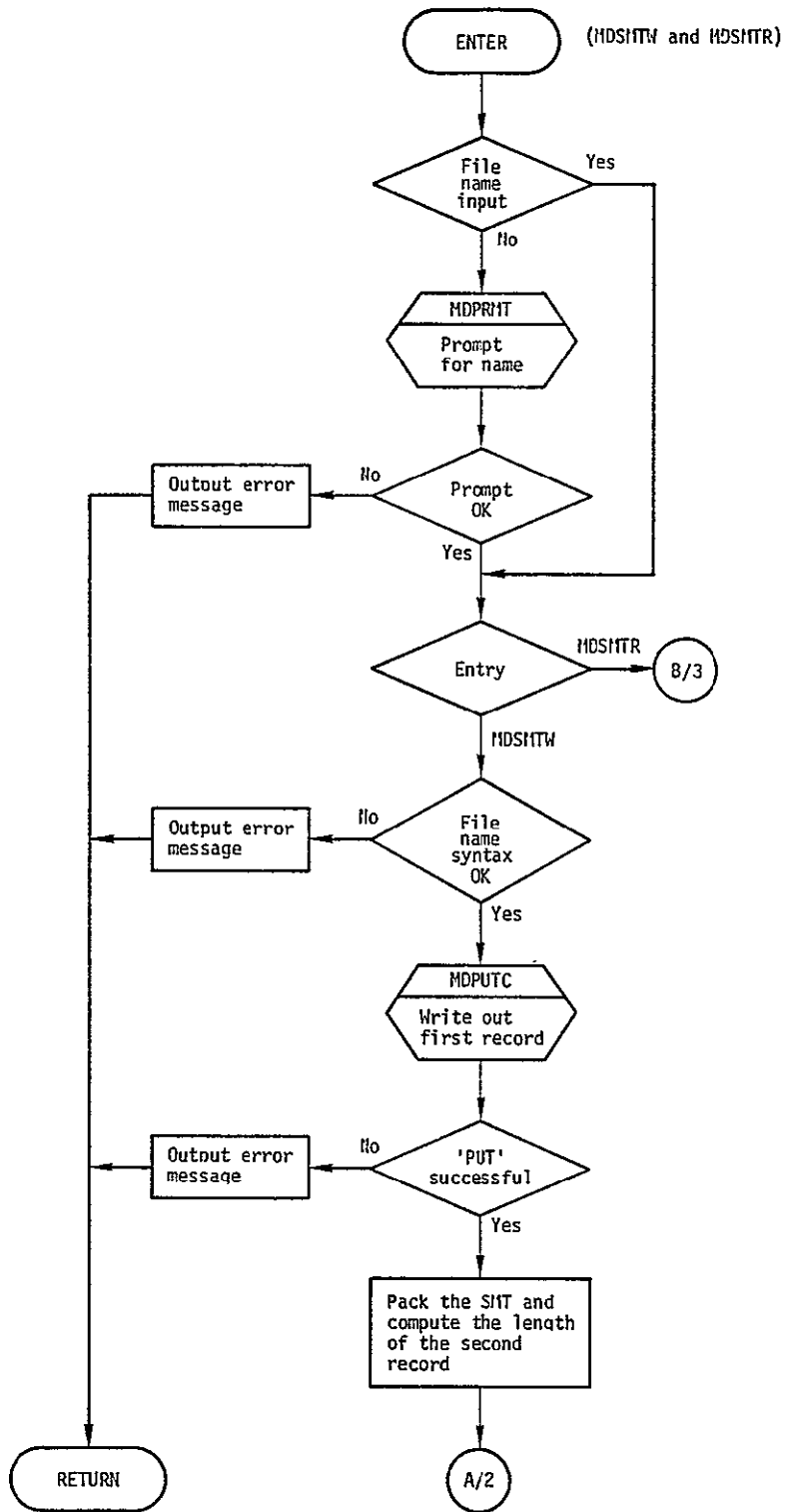
NAME	I
EOL	I

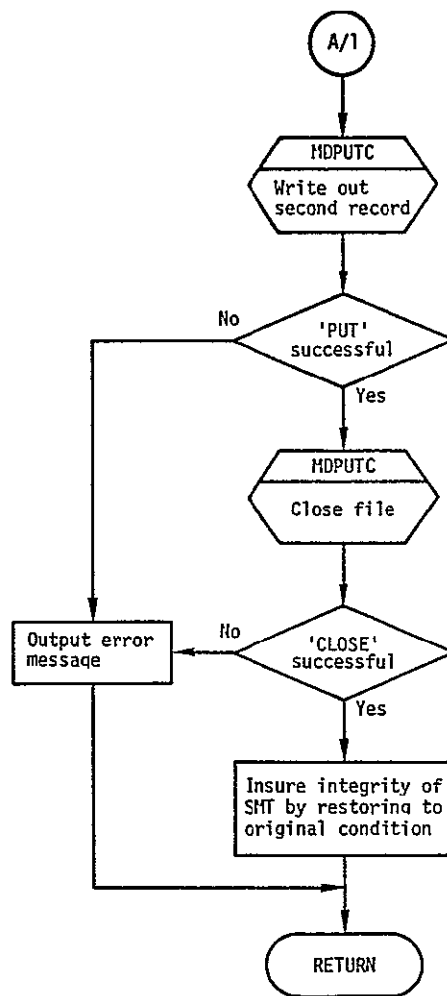
COMMON / MDBUFF /

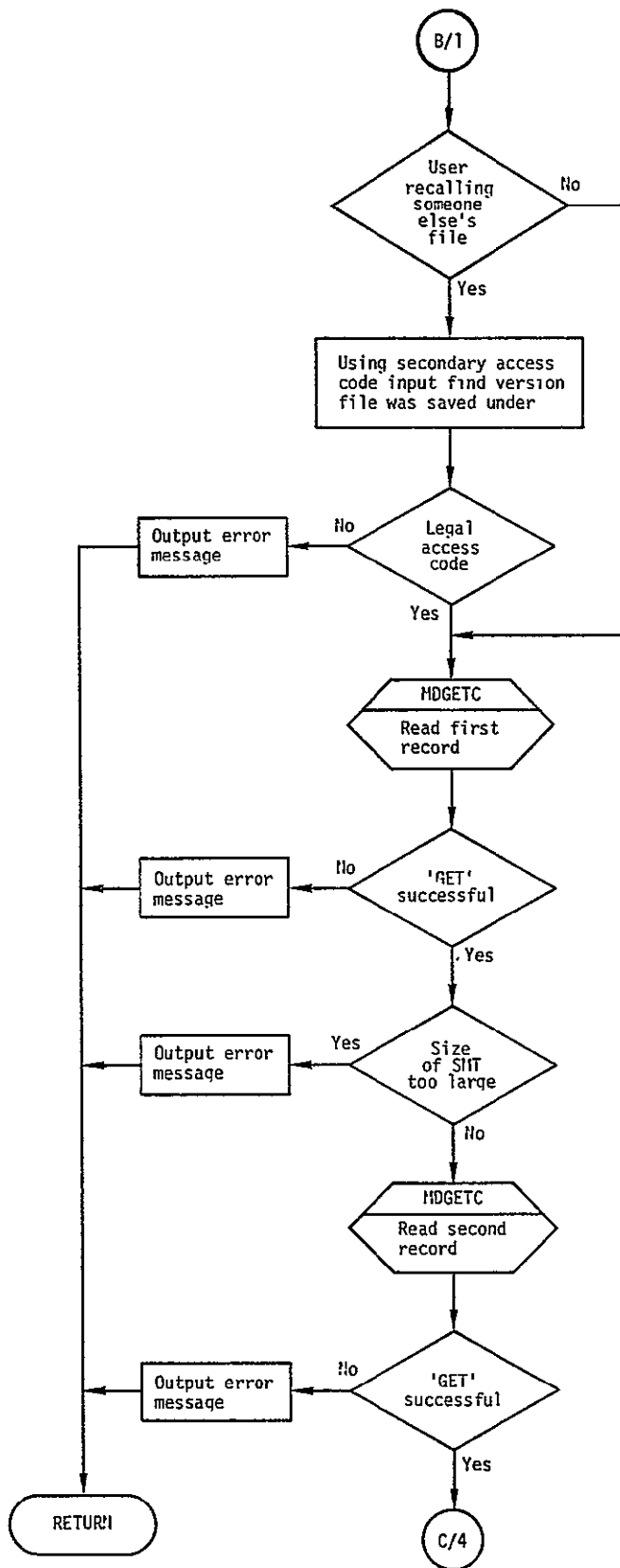
VARB 1/0

MDLEN	I
BDATA	I
OSIZE	I
WB	I

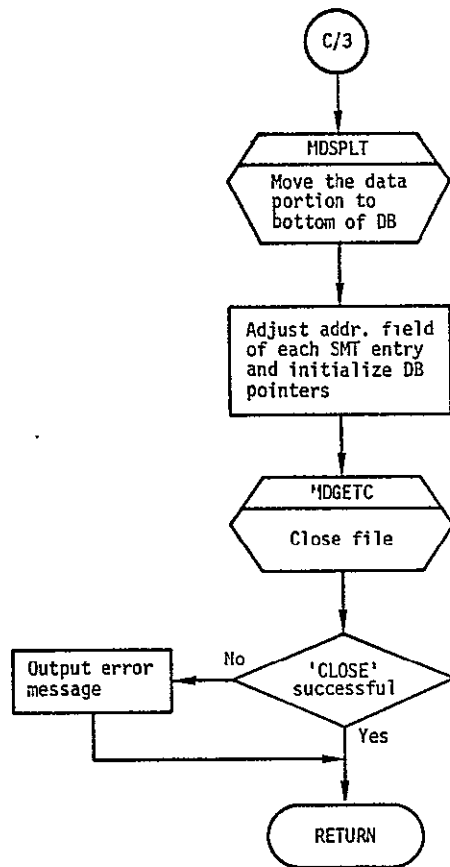
LOCAL COMMON







REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDSPLT - Utility Routine

MDSPLT will take a buffer and split it into two parts. It is primarily used by MDCNT to separate the argument specifications and the data and by MDCMT to separate the commands and the temporary edits.

Method

Input: All input to MDSPLT comes through the calling sequence and consists of a buffer containing the area to be split, the length (in words) of the area, the size (in words) of the array to be split and an index to the first word of the array to split off.

Processing: The buffer is separated into two parts. Any unused area is zeroed. The index to the array split off is calculated for output.

Output: All output is through the calling sequence and is the buffer containing the split array and an index which points to the first word of the array split off.

U SAGE

ENTRY MDSPLT
CALL MDSPLT (WB,MDLEN,SIZE,BDATA)

ARGMT	I/O	TYPE	DIM	DEFINITION
WB	I/O	I	VARB	BUFFER CONTAINING DATA TO BE SPLIT
MDLEN	I	I	1	SIZE (IN WORDS) OF WB
SIZE	I	I	1	SIZE (IN WORDS) OF THE PORTION OF DATA TO BE SPLIT AWAY
BDATA	I/O	I	1	INDEX TO THE PORTION OF THE DATA TO BE SPLIT AWAY

EXTERNAL REFERENCES
NONE

DIAGNOSTICS
NONE

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE

MDTOC - Utility

This routine will perform the TOC directive and, in doing so, generate a listing of the contents of the SMT.

Method

Input: There is no input to this routine.

Processing: Upon entry, the data areas are packed (MDPACK). If the SMT is empty, the user is informed of such and informed of the available size. Control is then returned to the submonitor (MDSMON).

If not empty, each SMT entry is listed. The list for each entry includes: the entry's alphanumeric name, its type, its size, its I-dimension, and its J-dimension. After all entries are listed, a message is printed informing the user of how large the SMT area is and how much of this is currently being used. Control is now returned to MDSMON.

Output: This routine has no output other than the user requested listing of the SMT.

Usage

ENTRY MDTOC
CALL MDTOC

EXTERNAL REFERENCES
MDPACK

DIAGNOSTICS
*** SMT EMPTY
THE SMT TO BE LISTED CONTAINS NO ENTRIES

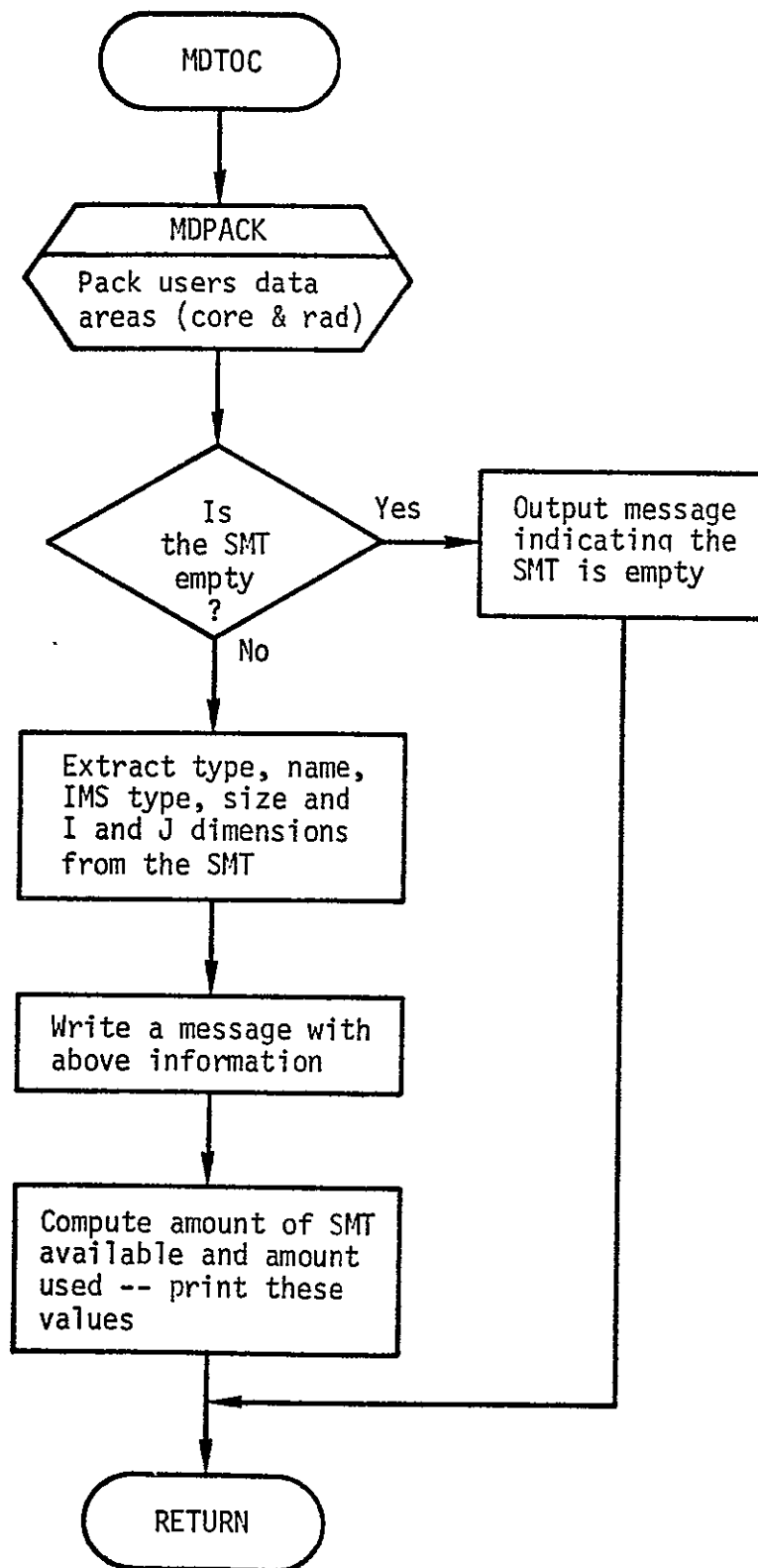
EXTERNAL STORAGE
NONE

BLANK COMMON
VARB I/O

DBADDR	I
DBMAX	I
DBSTRT	I
NTRY	I

LOCAL COMMON

NONE



MDTOC Functional Flow Diagram

MDUTIL - Utility Support Routine

MDUTIL performs the utility directives DUMP and DELETE. As more directives are implemented in the prototype, MDUTIL will take on the expanded role of performing them also.

Method

Input: All input is through the calling sequence and is: the buffer containing the directive and an indication of which directive has been entered.

Processing: If the DUMP directive has been entered, it is scanned for correct syntax and the presence of an optional type flag. If the flag is present the data is listed by this type. If not present, the data is listed by its internal type. After performing the dump, control is returned to MDSMON.

If the DELETE directive has been entered, the syntax is verified and the data area deleted. The user is informed of a successful deletion or of the fact the area does not exist. In either case, control is returned to MDSMON.

Output: There is no output from MDUTIL other than what the user obtains by doing the directive.

USAGE

```
ENTRY MDUTIL
  CALL MDUTIL(DIRECT,INPUT)
```

ARGMT	I/O	TYPE	DIM	DEFINITION
DIRECT	I	I	I	NUMERICAL VALUE INDICATING THE DIRECTIVE JUST ENTERED
INPUT	I	I	VARB	BUFFER CONTAINING THE DIRECTIVE

EXTERNAL REFERENCES

```
MDELET
MDGET
MDLIST
```

DIAGNOSTICS

```
*** COULD NOT FIND .....
      THE SPECIFIED ARRAY TO BE DUMPED COULD NOT BE FOUND
      IN THE SMT
*** COULD NOT READ .....
      THE ARRAY TO BE DUMPED COULD NOT BE READ
*** INVALID SYNTAX
      THE DIRECTIVE CONTAINED A SYNTAX ERROR
MDUTIL EXECUTED
      A DIRECIVE NOT YET IMPLEMENTED IN THE MONITOR HAS
      BEEN REQUESTED. CONTROL IS RETURNED TO MDSMON WITH
      NO ACTION TAKEN.
```

EXTERNAL STORAGE

```
NONE
```

BLANK COMMON

```
NONE
```

COMMON / MDCODE /

```
VARB      I/O
```

```
NAME      I
UPPARW    I
COMMA     I
```

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

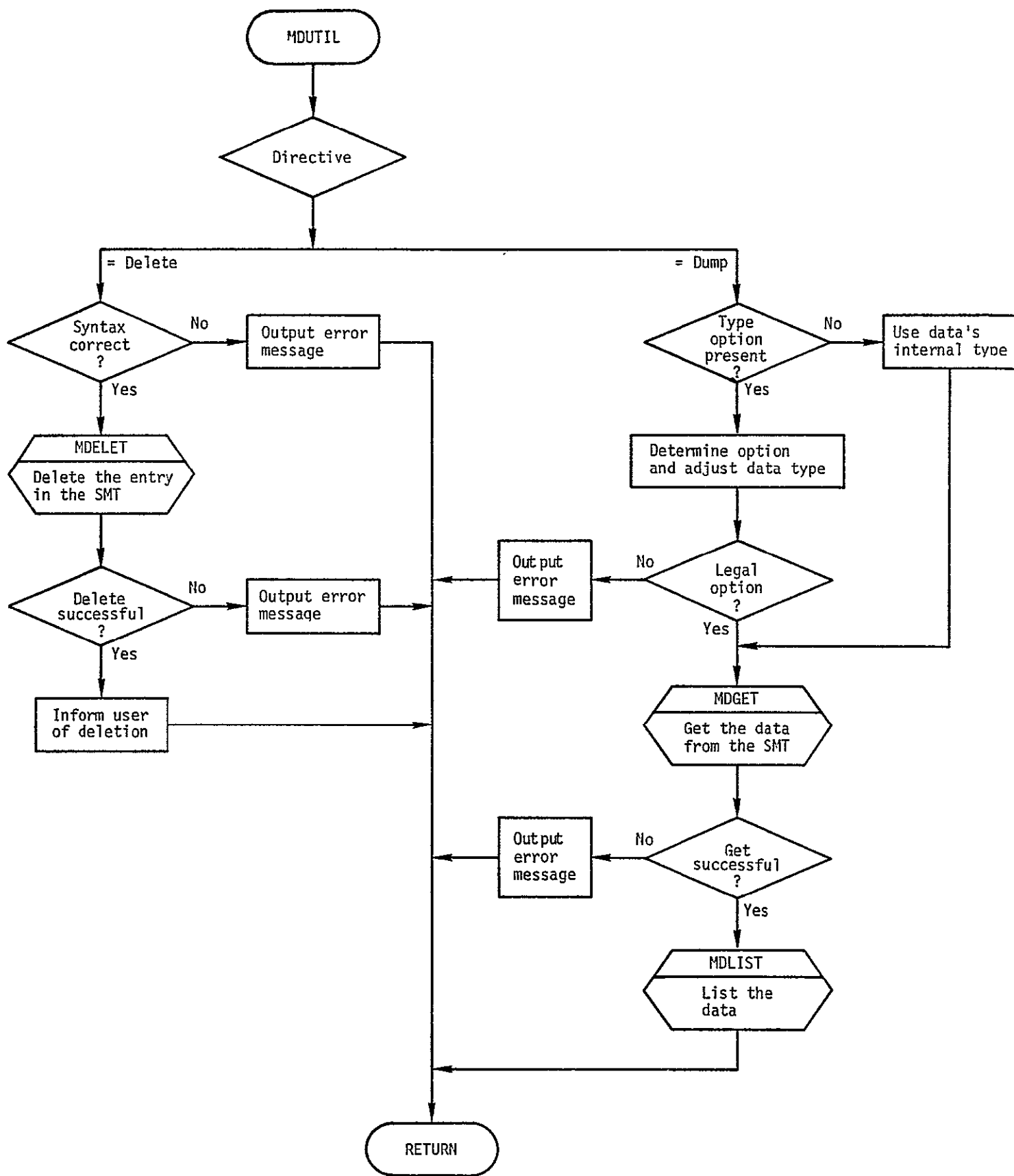
COMMON / MDBUFF /

```
VARB      I/O
```

```
MDLEN     I
WB
```

LOCAL COMMON

```
NONE
```



MDUTIL Flow Diagram

SEARCH - Binary Search Routine

SEARCH performs an examination of an ordered (sorted) input array to detect the presence of a specified entry. Examination of multiple rows of the array on a prioritized basis is provided.

Method

Input: SEARCH accepts as input an ordered array of data and a column, or item, to be compared to the columns of the array. The comparison is based on a prioritized set of search keys also input.

Processing: The technique used to examine the input array is a binary search, also known as search by bisection.

Output: The column number of a match or a flag indicating no match is returned.

USAGE

ENTRY SEARCH

CALL SEARCH (TAB, ID, JD, NKEY, KEY, FIND, LOC)

ARGMT	I/O	TYPE	DIM	DEFINITION
TAB	I	I, H	ID, JD	ARRAY SORTED BY KEY ROWS TO BE SEARCHED IN THOSE ROWS FOR THE SPECIFIED ENTRY
ID	I	I	1	NUMBER OF ROWS IN TAB AND LENGTH OF FIND
JD	I	I	1	NUMBER OF COLUMNS IN TAB
NKEY	I	I	1	NUMBER OF SEARCH KEYS IN KEY
KEY	I	I	NKEY	SEARCH KEYS. ROW NUMBERS OF ROWS OF TAB AND ENTRIES OF FIND TO BE COMPARED (PREFIXED WITH MINUS SIGN TO DESIGNATE ALPHABETIC COMPARISON). THE SEQUENCE OF VALUES IN KEY ESTABLISHES THE SEARCHING PRIORITY, I.E., KEY(1) INDICATES THE PRIMARY, KEY(2) THE MAJOR, ETC.
FIND	I	I, H	ID	COLUMN TO BE COMPARED TO COLUMNS OF TAB IN THE KEY ENTRIES
LOC	O	I	1	COLUMN NUMBER OF TAB MATCHING FIND IN THE KEY ENTRIES OR ZERO IF NOT FOUND

EXTERNAL REFERENCES

NONE

RESTRICTIONS

THE INPUT ARRAY OF MUST BE ALGEBRAICALLY AND/OR ALPHABETICALLY ORDERED IN THE KEY ROWS TO BE SEARCHED.

THE ABSOLUTE VALUES OF THE SEARCH KEYS MUST CORRESPOND TO ROW NUMBERS OF THE INPUT ARRAY

DIAGNOSTICS

IMPROPER VALUE FOR SEARCH POINTER
THE ABSOLUTE VALUE OF THE INDICATED ELEMENT OF KEY IS ZERO.

EXTERNAL STORAGE

NONE

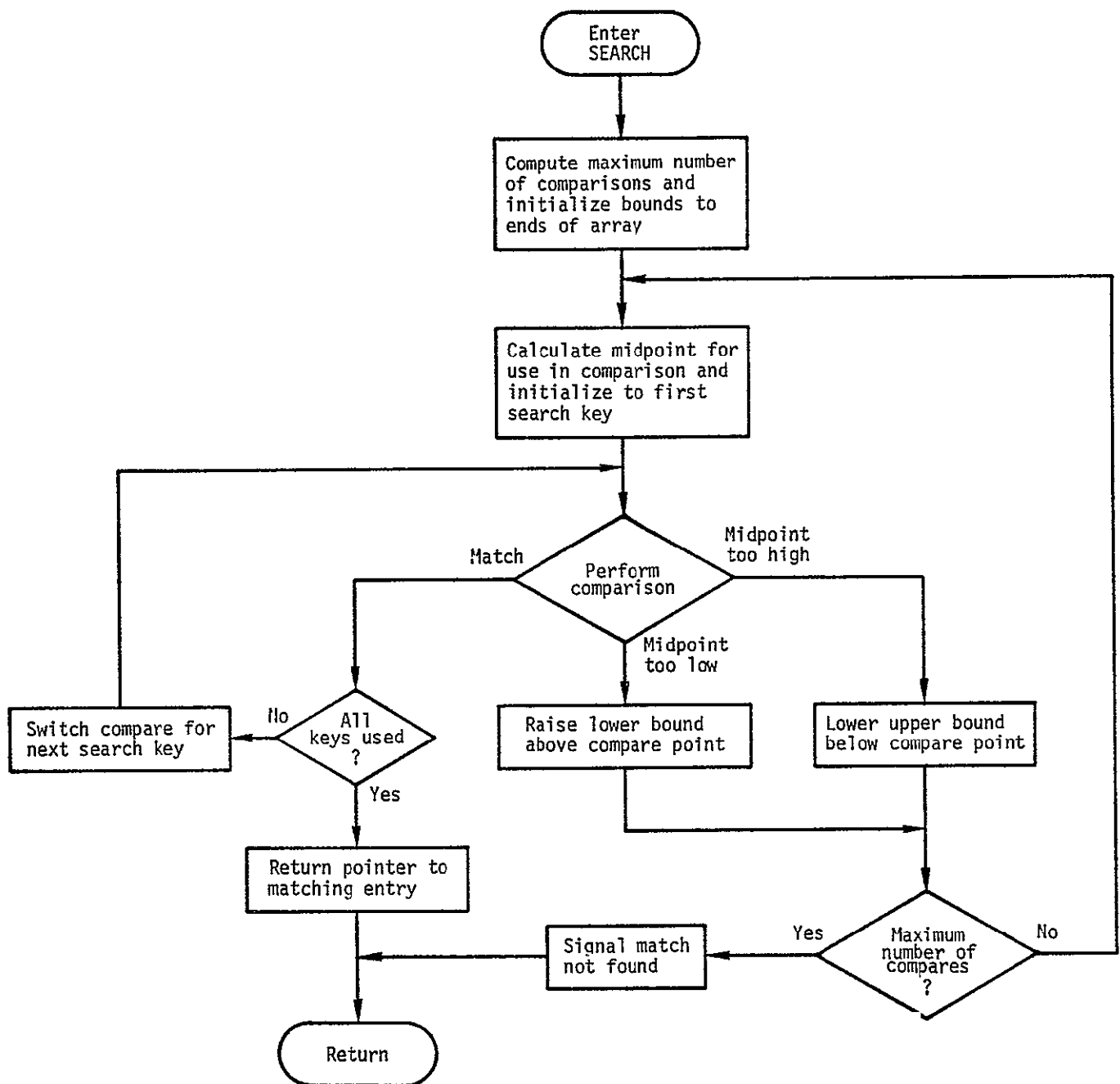
BLANK COMMON

NONE

LOCAL COMMON

NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



SORT2 - Array Sorting Package

SORT2 and SORT1 provide very fast algebraic and/or alphabetic sorting of arrays based on any number of sort keys. The sorting is based on the contents of specified rows of the input array.

Method

Input: The primary inputs are the array to be sorted and an array of sort keys designating the rows on which to base the sort, their priority (order of sorting) and which are to be sorted algebraically and which alphabetically. The size of the array and number of sort keys is also specified. Entry point SORT2 further provides for the parallel manipulation of an additional array of data during the sorting of the input array, assuming a relationship exists between the entries of the two arrays.

Processing: The sorting algorithm is a variation of a splitting technique described by R. C. Singleton, Communication of the ACM, Volume 12/Number 3/ March 1969, p. 85. SORT2 is an extension and generalized implementation of the technique.

The method is the sort analogy to a binary search. First the array is split and reorganized such that all "low" values are placed in the top half of the array and all "large" values in the bottom. Indices bounding the bottom are then saved. The top half is then split and again all "low" values are moved to the top and "large" values to the bottom. Again the indices of the bottom are saved. The process is continued until a top to be split contains no more than three values. These are arranged in order and splitting continues by retrieving a bottom section from the index queue on a last in first out basis. Queue space for the bottom indices is related to the number of entries by the expression

$$m = 2^{Q+2} + 2^Q - 1$$

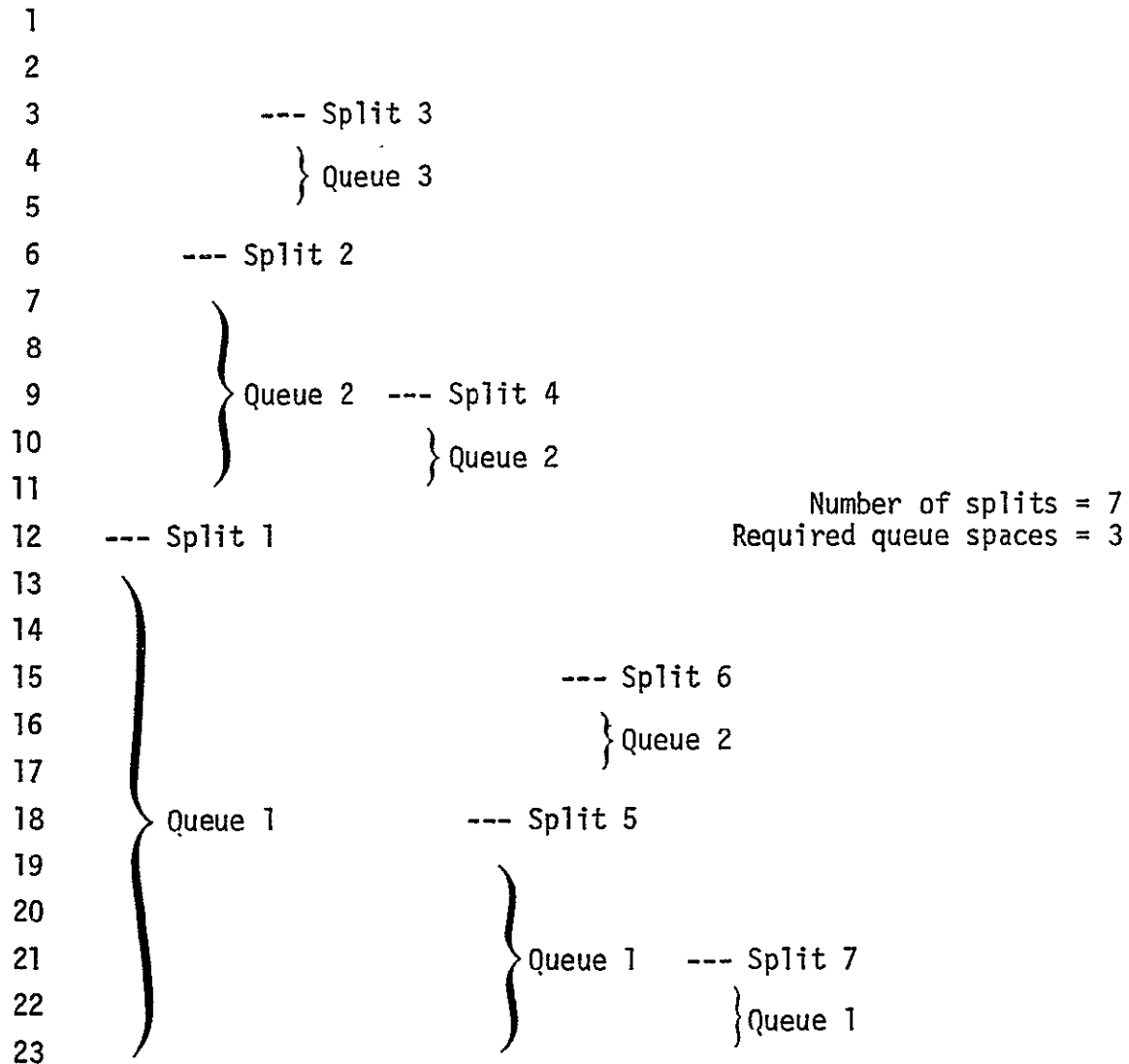
where

m is the maximum number of entries to be sorted

Q is the number of index pairs for which storage is provided

Q is 14 in the present implementation which permits an array of up to 81,919 columns to be sorted.

The technique is illustrated in the following example and the accompanying flow diagram.



Example of Splitting and Queueing an Array of 23 Entries

Output: The output from SORT1 and SORT2 is the sorted input array. SORT2 also outputs the associated array which was operated on in parallel with the primary array.

USAGE

ENTRY SORT2

CALL SORT2 (AI, IA, J, NKEYS, KEYS, BI, IB, AO, BO)

ARGMT	I/O	TYPE	DIM	DEFINITION
AI	I	I,H	IA,J	ARRAY TO BE SORTED ACCORDING TO THE DATA IN KEYS
IA	I	I	1	NUMBER OF ROWS IN AI AND AO
J	I	I	1	NUMBER OF COLUMNS IN AI, BI, AO AND BO
NKEYS	I	I	1	NUMBER OF SORT KEYS IN KEYS
KEYS	I	I	NKEYS	SORT KEYS. ROW NUMBERS OF ROWS OF AI ON WHICH TO SORT (PREFIXED WITH MINUS SIGN TO DESIGNATE ALPHABETIC SORT). THE SEQUENCE OF VALUES IN KEYS ESTABLISHES THE SORTING PRIORITY, IE., KEYS(1) INDICATES THE PRIMARY, KEYS(2) THE MAJOR, ETC.
BI	I	FREE	IB,J	ARRAY TO BE OPERATED ON IN PARALLEL WITH AI
IB	I	I	1	NUMBER OF ROWS IN BI AND BO
AO	O	I,H	IA,J	SORTED ARRAY AI
BO	O	FREE	IB,J	SORTED ARRAY BI

ENTRY SORT1

CALL SORT1 (AI, IA, J, NKEYS, KEYS, AO)

ARGMT	I/O	TYPE	DIM	DEFINITION
AI	I	I,H	IA,J	ARRAY TO BE SORTED ACCORDING TO THE DATA IN KEYS
IA	I	I	1	NUMBER OF ROWS IN AI AND AO
J	I	I	1	NUMBER OF COLUMNS IN AI AND AO
NKEYS	I	I	1	NUMBER OF SORT KEYS IN KEYS
KEYS	I	I	NKEYS	SORT KEYS. ROW NUMBERS OF ROWS OF AI ON WHICH TO SORT (PREFIXED WITH MINUS SIGN TO DESIGNATE ALPHABETIC SORT). THE SEQUENCE OF VALUES IN KEYS ESTABLISHES THE SORTING PRIORITY, IE., KEYS(1) INDICATES THE PRIMARY KEYS(2) THE MAJOR, ETC.
AO	O	I,H	IA,J	SORTED ARRAY AI

EXTERNAL REFERENCES
NONE

RESTRICTIONS

THE ABSOLUTE VALUES OF KEYS MUST BE BETWEEN 1 AND J INCLUSIVE
THE MAXIMUM NUMBER OF COLUMNS (VALUE OF J) WHICH CAN BE
ACCOMMODATED IS PRESENTLY DEFINE AS 40959.

DIAGNOSTICS

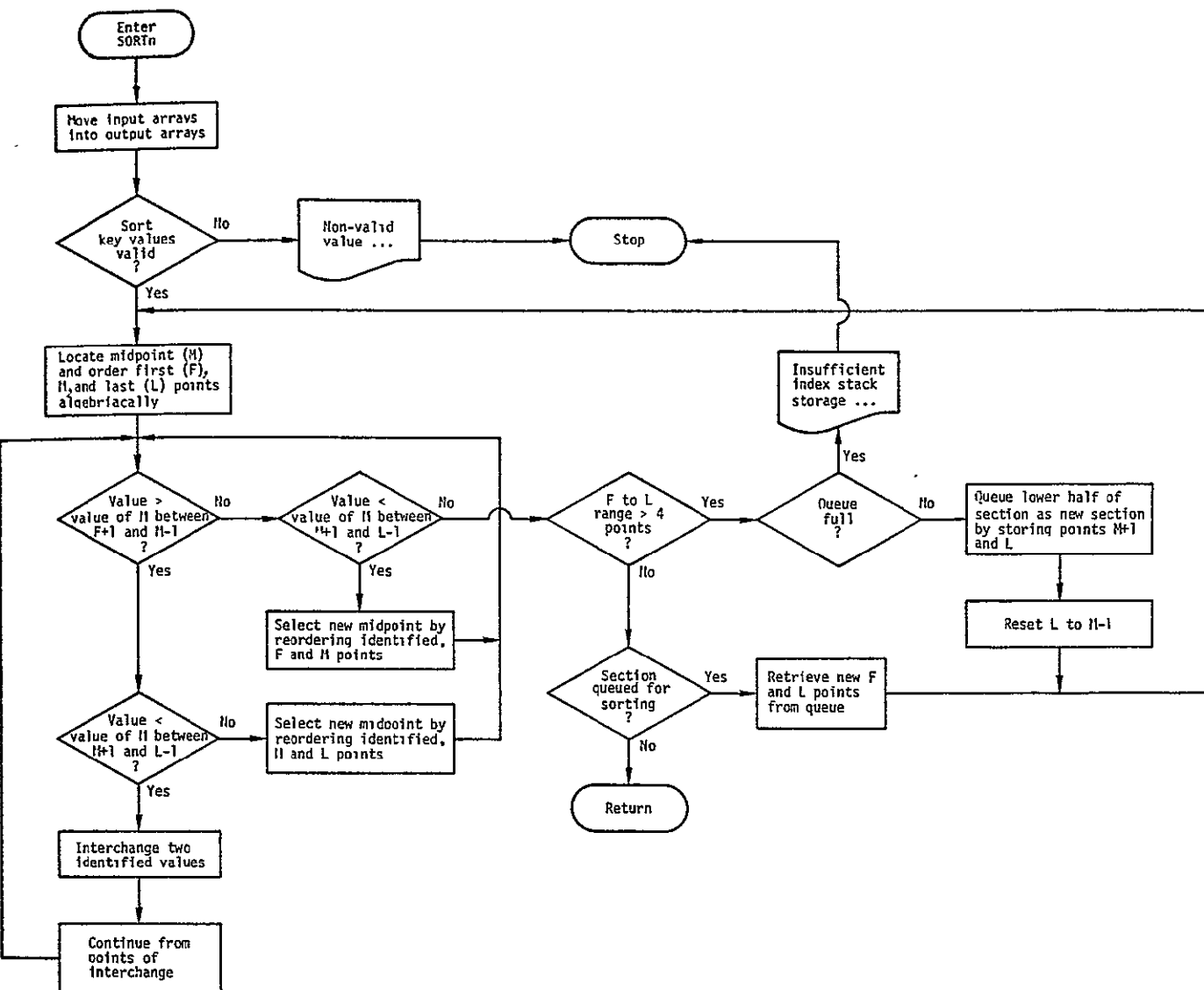
INSUFFICIENT INDEX STACK STORAGE FOR SORTING ELEMENTS
AS PRESENTLY CONFIGURED A QUEUE FOR SAVING SECTIONS TO
BE SORTED WILL ONLY ACCOMMODATE 40959 COLUMNS OF AI.
NON-VALID VALUE FOR SORT KEY
THE ABSOLUTE VALUES OF THE SORT KEYS MUST CORRESPOND TO

THE ROW NUMBERS OF A1.

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE



SORT1 and SORT2 Functional Flow Diagram
8.17-6

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

DCTMOD - Library Maintenance

DCTMOD is a stand alone Fortran program to delete processors from the MDAS catalog and to modify default control tables of processors in the MDAS catalog.

Method

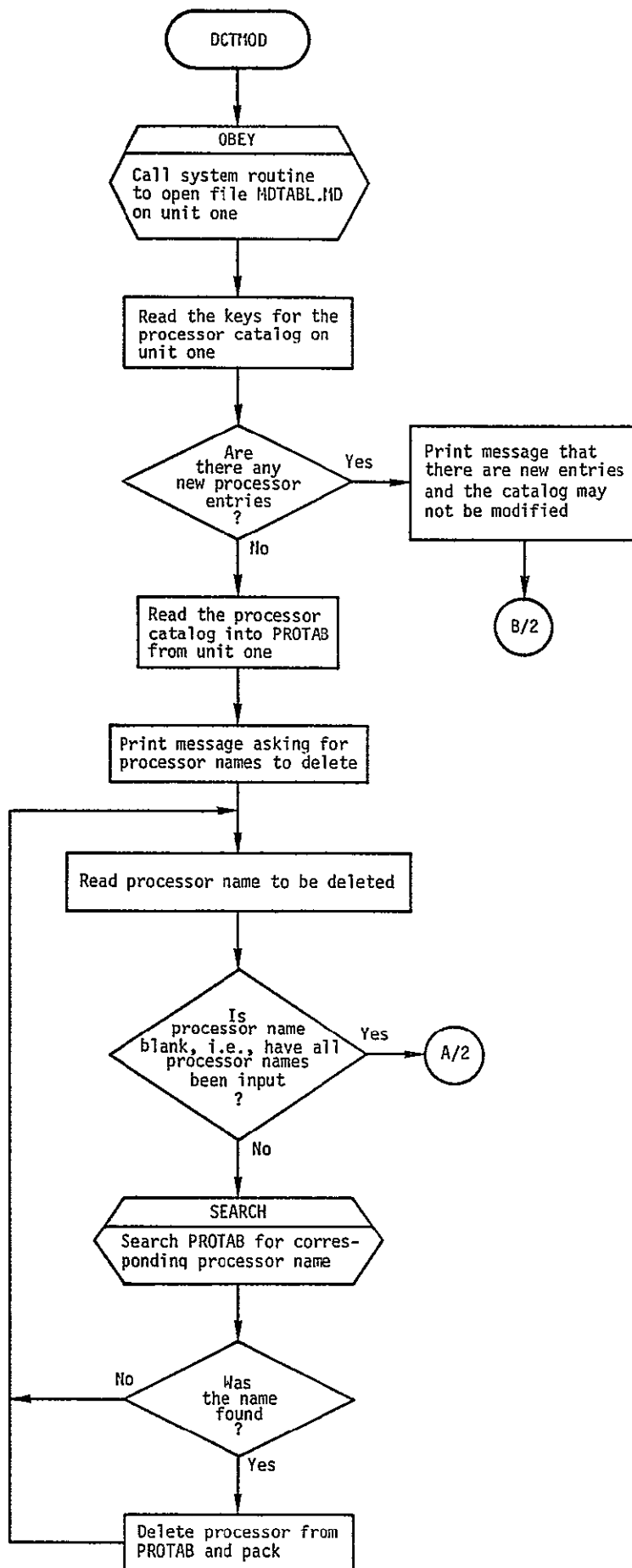
Input: The processor catalog file (MDTABL.MD) to have modifications and deletions must reside on unit one.

Processing: When executed, DCTMOD brings into memory the processor catalog file from unit one. The INFONET system routine OBEY is used to open the file and equate the file MDTABL.MD to unit 1. If there are new processor entries indicated by the catalog key, a message is printed stating that the catalog may not be modified. If there are new processor entries, they must be put in the catalog previous to a DCTMOD execution.

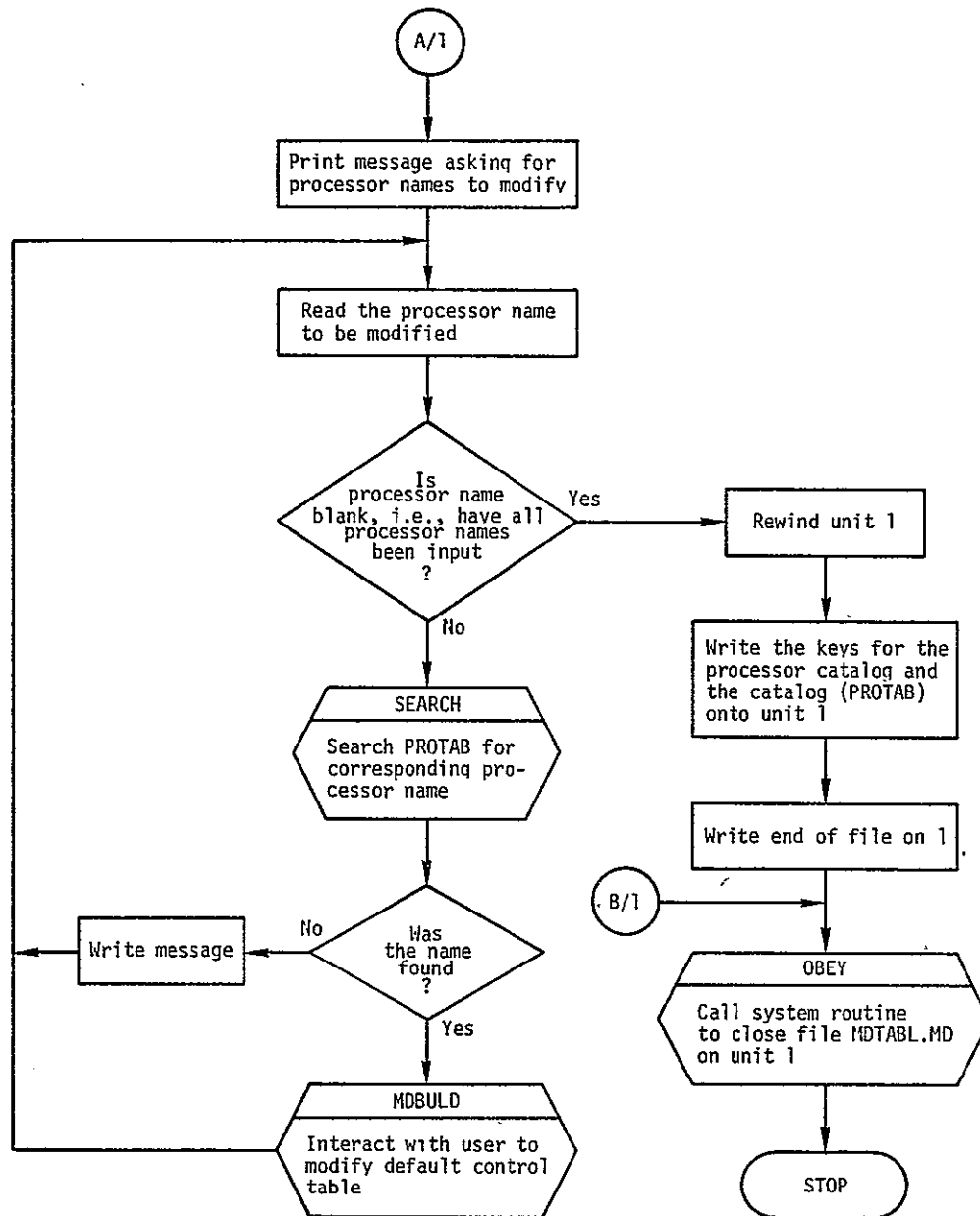
After the catalog is in memory, the user is prompted for each processor to be deleted. For each name input, a search of the catalog (PROTAB) is made. If no entry was found, the user is requested to input another name. Otherwise the PROTAB entry corresponding to the name is deleted and the table is then packed.

After the user has indicated by pressing the carriage return that all desired processors have been deleted, the user is prompted for each processor name to modify the default control table. For each name input, a search of PROTAB is made. If no entry was found, a message is printed and the user is requested to input another name. Otherwise, MDBULD is called to interface with the user in modifying the default control table. After the user has indicated that all desired processor default control tables have been modified, DCTMOD then writes the updated processor catalog back to unit one. The INFONET routine OBEY is used to close the file MDTABL.MD equated to unit one. Execution of DCTMOD is then terminated.

Output: The update processor catalog file (MDTABL.MD) will be placed back on unit one.



DCTMOD Flow Diagram



DCT/MD Flow Diagram (Continued)

9.1-4

Page 2 of 2

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDADDR - Library Maintenance Processor

MDADDR is an extension of the library maintenance programs MDGENR, DCTMOD and MDUMPC. Its execution is triggered by the boot logic when changes to the library are detected during catalog loading (see Boot Logic (MDAS)). MDADDR integrates these changes into the library catalog and produces an updated catalog.

Method

Input: The primary inputs to MDADDR are the library catalog and catalog control keys as loaded by the boot logic. The control keys indicate the original catalog prior to the library maintenance activities and the area containing new processor data. Also input are the swap area sizes and origin addresses of the SMT and ephemeris buffer.

Processing: MDADDR and its associated routines are designed for use by subsystem maintenance personnel, therefore special log-on access codes are required in order to proceed with the library maintenance process. Each new catalog entry is examined in turn to determine whether or not it refers to the submonitor (MDSMON) or MDADDR itself. As shown in Figure 2 of Appendix C these data share the first catalog entry and thus are handled separately from other catalog entries.

Processor entries are compared with the list of existing cataloged processors and the default control table maintenance routine MDBULD is invoked to build or modify a default control table. The catalog data for the processor is then moved to an appropriate location in the catalog which is then re-sorted alphabetically, if necessary.

These procedures result in reduction in the amount of memory occupied by the catalog as new data for existing processors are moved into the catalog. To maximize SMT size the origins of the ephemeris buffer and SMT are adjusted to utilize the vacated area.

The finalized catalog is output to the file MDTABL.MD destroying the previous catalog. Library maintenance thus completed, MDADDR verifies the adequacy of the swap area for loading the submonitor and returns to the resident.

Output: MDADDR outputs the updated catalog to memory and mass storage and adjust the origins of the ephemeris buffer and SMT as appropriate.

USAGE

ENTRY MDADDR
CALL MDADDR

EXTERNAL REFERENCES

MDBULD, MDLOGO, SEARCH, SORT1

DIAGNOSTICS

EXTENT OF MD\$MON (.....) IS TOO LARGE FOR CURRENT MDAS
CONFIGURATION (.....)

THE LOAD MODULE OF THE SUBMONITOR REQUIRES A SWAP AREA
LARGER THAN THE ALLOCATED REGIONS. DETERMINE THE
REQUIREMENTS, EDIT MDAS=PNC TO REVISE THE VALUES IRES
AND DRES APPROPRIATELY AND REASSEMBLE AND LINK MDAS.

MDAS IS TEMPORARILY UNAVAILABLE DUE TO MAINTENANCE ACTIVITIES
PLEASE TRY AGAIN LATER

SINCE THE CONSTRUCTION AND MODIFICATION OF DEFAULT
CONTROL TABLES IS NOT CONSIDERED A USER ACTIVITY,
MDADDR PROHIBITS USER ACCESS TO MDAS UNTIL ALL SUCH
ACTIVITIES HAVE BEEN COMPLETED.

MDAS LIBRARY CONTROL TABLE UPDATED
THIS MESSAGE SIGNALS THE COMPLETION OF THE CATALOG
MAINTENANCE PROCESS.

EXTERNAL STORAGE

THE MERGED CATALOG IS WRITTEN TO FILE MDTABL.MD

BLANK COMMON

VARB I/O

BADGE I

DSIZE I

DBSTRT I/O

ISIZE I

IEPHST I/O

PROTAB I/O

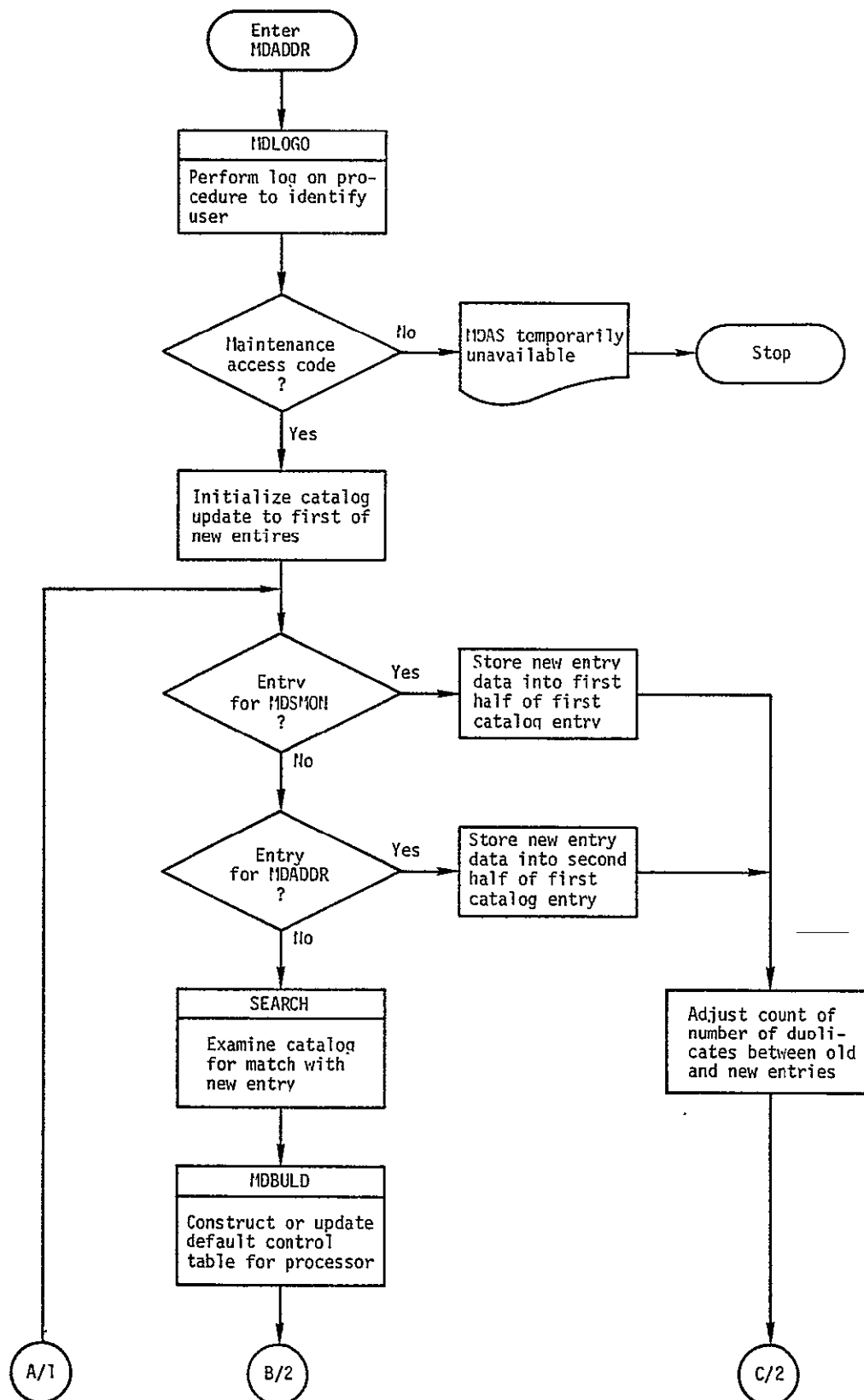
PTABKY I/O

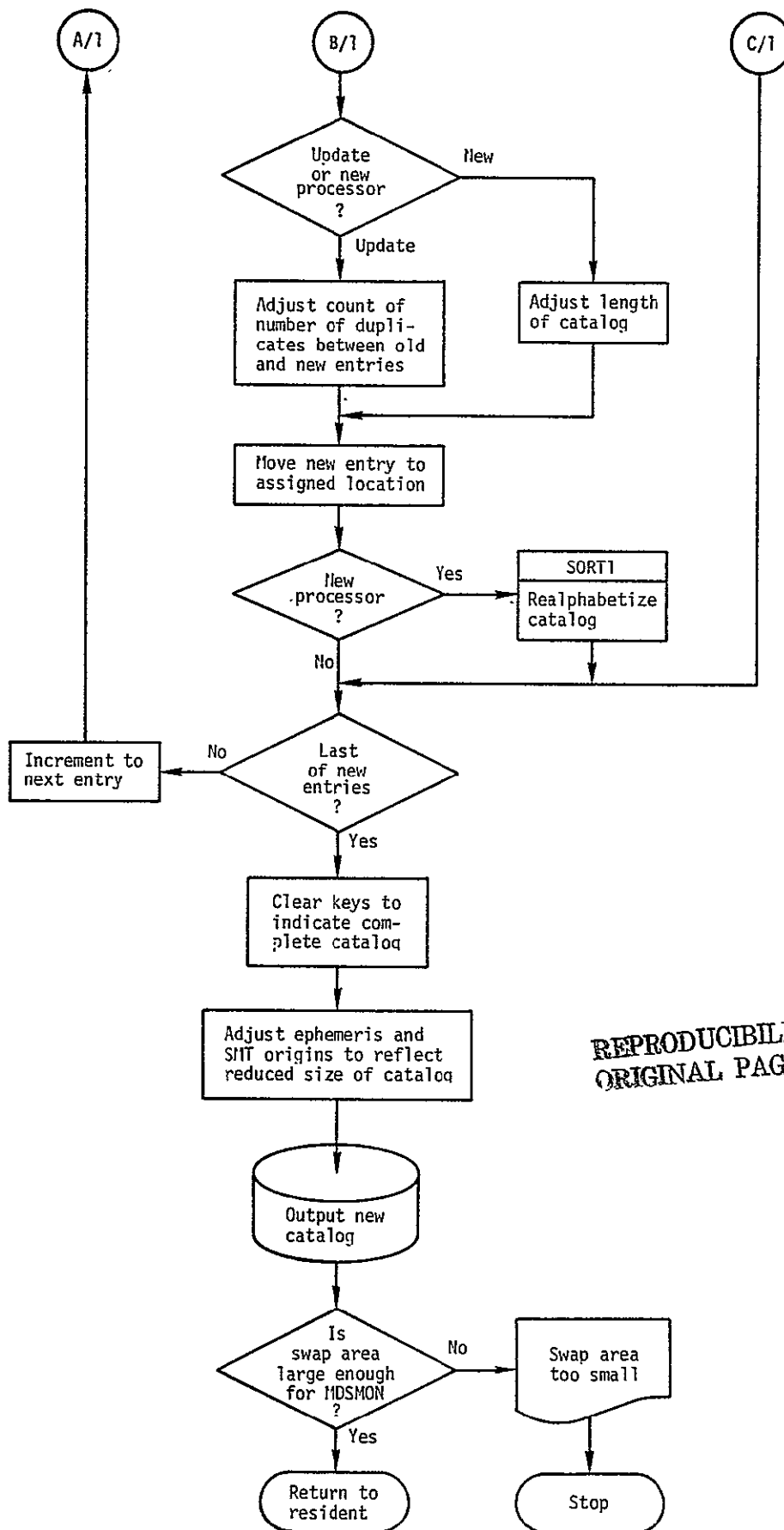
LOCAL COMMON / MDBUFF /

VARB	I/O	TYPE	DIM	LOC	RELADD	DEFINITION
------	-----	------	-----	-----	--------	------------

MDLEN	0					
-------	---	--	--	--	--	--

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR





REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDBULD - Library Maintenance

The purpose of MDBULD is to build a default control table for the processor requested. It also updates the default control table length, argument definition length, and processor revision number in the PROTAB.

Method

Input: The input to MDBULD consists of the entry number of the processor in the PROTAB (system directory) and flag designating whether this processor is a new or existing one. These inputs are passed through the calling sequence.

Processing: MDBULD, if the default control table does not need modification, moves the old PROTAB default control table length, argument definition length, and revision number into the new PROTAB entry for the processor. If the request was for a modification, SEARCH is called to find the processor in the PROTAB and the values updated.

If a modification was requested for the default control table, then MDGETC is called to get the existing default control table for the requested processor. By a series of calls to MDPRMT, the user is prompted for the information needed to update or build the argument specifications portion of the default control table. The user may input the following parameters for the default control table: the revision number, number of arguments, and scan flag for the label field, and the argument identification, I-dimension, J-dimension, type, constant argument and I/O flag for each argument. If the user does not input a value, the value will not be changed or reinitialized. The data completion and argument completion flags are set in the control table according to the status of the data.

If there is a request to list the table, MDLIST is called to display this data. Since the argument specifications and the data are packed together when residing on disc, MDSPLT is called to separate them for storage in the working buffer. After the control table has been updated or built, if there is a request to edit it, MDEDCN is called. If no further modifications are desired, MDCTPK is called to pack the argument specifications together with the data before MDPUTC is called to write this information on disc.

Output: The output from MDBULD is an updated PROTAB and an updated or newly built default control table.

USAGE

ENTRY MDBULD

CALL MDBULD (KEY,FLAG)

ARGMT	I/O	TYPE	DIM
-------	-----	------	-----

DEFINITION

KEY	I	I	I
-----	---	---	---

KEY IS THE PROTAB ENTRY NUMBER OF THE PROCESSOR

FLAG	I	I	I
------	---	---	---

FLAG=0 INDICATES THIS IS A NEW PROCESSOR, FLAG NOT EQUAL 0 INDICATES THIS IS AN EXISTING PROCESSOR

EXTERNAL REFERENCES

MDCTPK

MDEDCN

MDGETC

MDLIST

MDPRMT

MDPUTC

MDSPLT

SEARCH

DIAGNOSTICS

***** IS A DUPLICATE IDENTIFIER

A DUPLICATE ARGUMENT IDENTIFIER WAS FOUND

*** READ ERROR WHILE READING RESPONSE

AN ERROR OCCURRED WHILE PROMPTING

*** SESSION CONCLUDED -- NO DEFAULT CONTROL TABLE GENERATED

AN ERROR OCCURRED WHILE PROCESSING THIS DEFAULT

CONTROL TABLE-NO DEFAULT CONTROL TABLE IS GENERATED

EXTERNAL STORAGE

NONE

BLANK COMMON

VARB I/O

PROTAB I/O

PTBLN I

COMMON /MDCODE/

EOS I

INTEG I

NAME I

COMMON /MDBUFF/

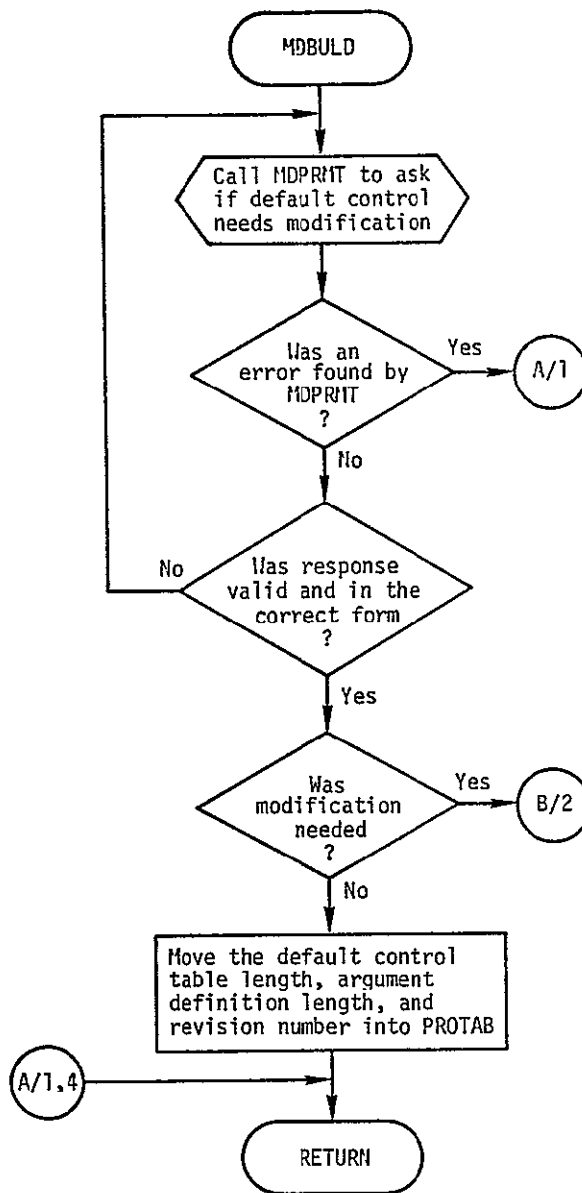
BDATA 0

DSIZE 0

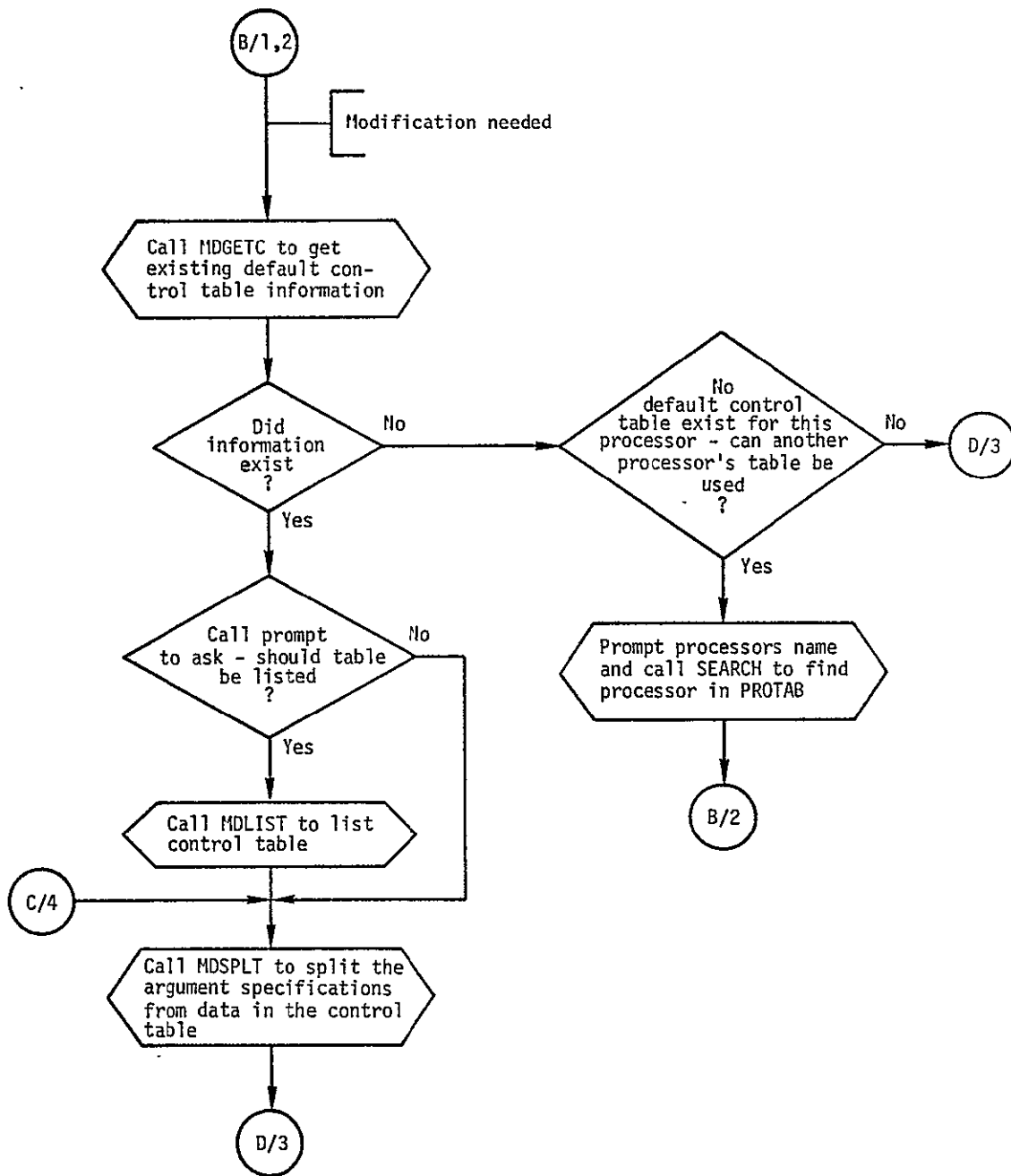
MDLEN 1

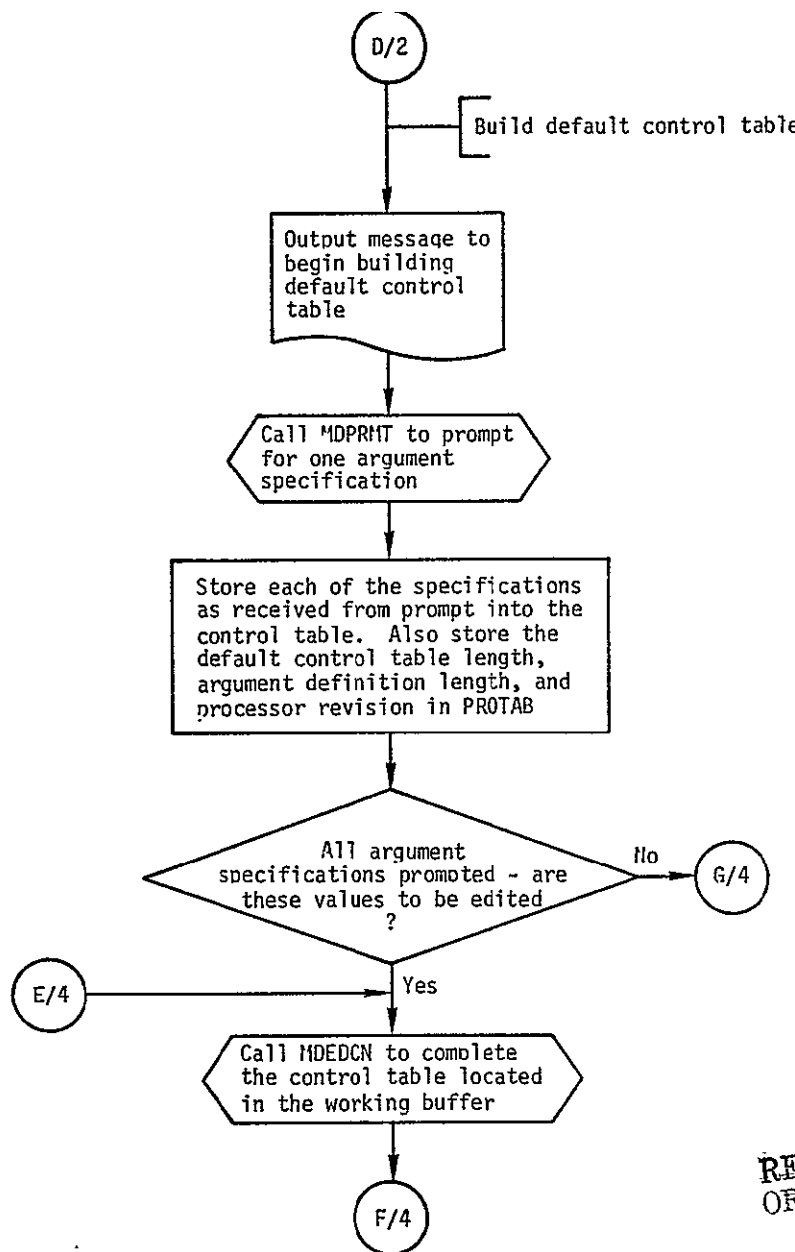
WB 0

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

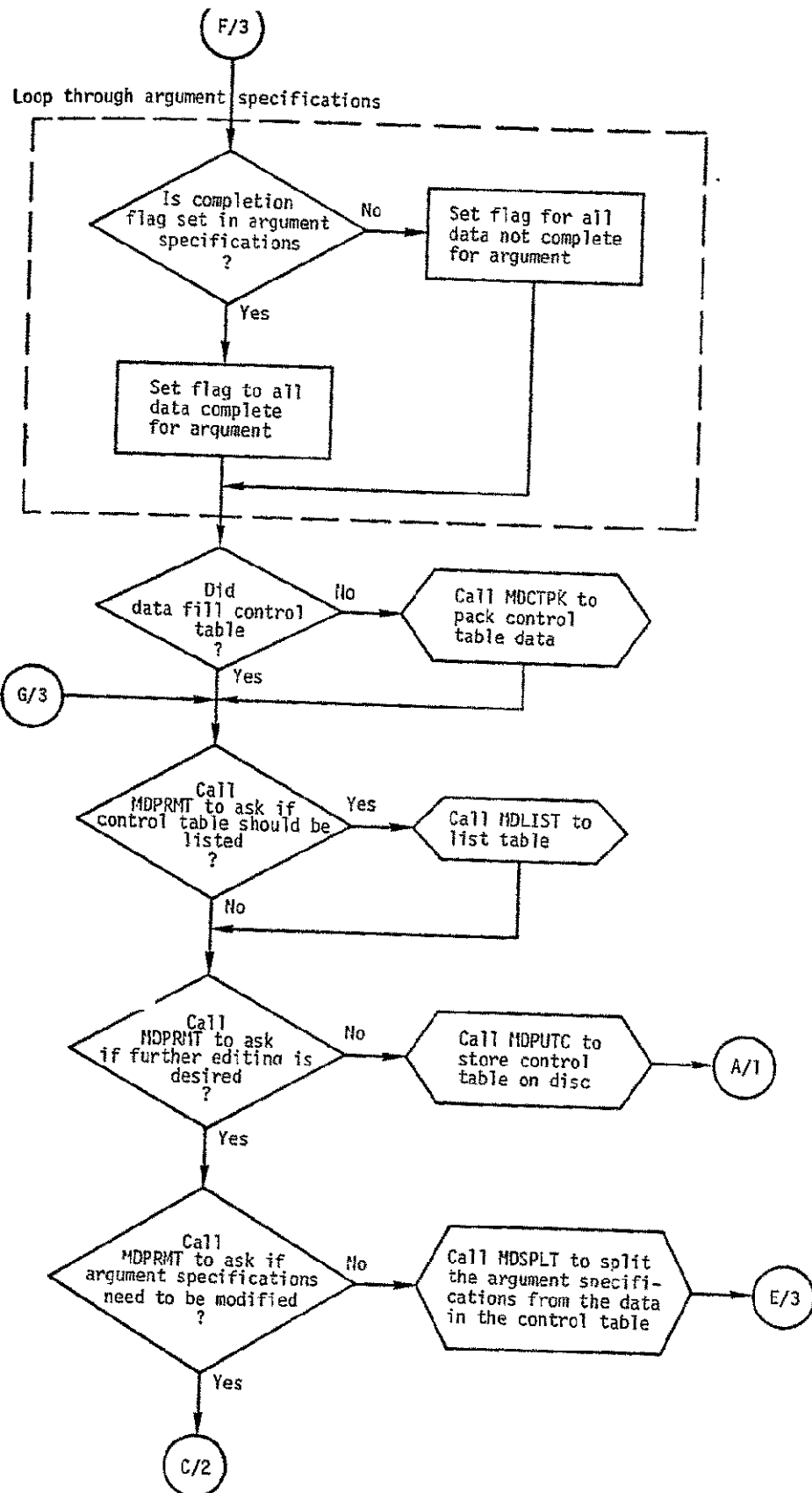


MDBULD Flow Diagram





REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDGENR - Library Maintenance

MDGENR is used to add a processor to the library or replace an old version of a processor with a new one. It records the appropriate information in the library catalog and generates a file of absolute code for use by the MDAS resident in loading the processor.

Method

Input: At the time of execution MDGENR is complete and requires no additional data for updating the library. It does read the catalog key record (first record) in order to update it with information pertinent to the new processor absolute file being generated.

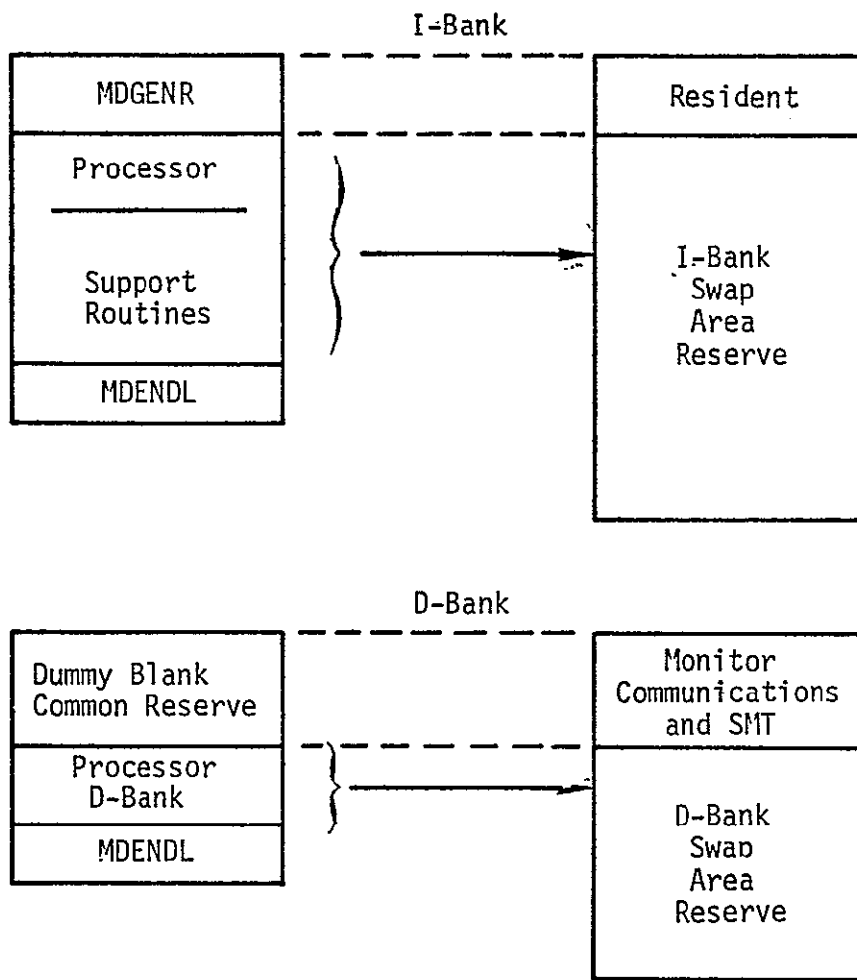
Data relative to the processor involved is assembled into MDGENR (see USAGE).

Processing: The name and entry point of the appropriate processor are assembled into MDGENR followed by a link edit which results in a load module headed by MDGENR with the processor and all supporting routines assigned to specific memory locations. MDGENR is precisely the size of the resident I-bank, thus the region assigned to the linked processor corresponds to the area in the I- and D-banks reserved for the swap area by the resident. The following figure illustrates the correspondence between MDGENR and MDAS memory allocations.

Externally defined symbols in MDENDL are used to determine the extents of the processor. This data is used to compute the lengths of the processor areas which together with beginning addresses are written to the catalog (see Figure C-2). The catalog key record is updated with the number of presently defined new library entries and a pointer to the entry corresponding to a new version of the MDADDR maintenance processor if one exists.

A new file is output containing two records. The file, named with the processor name and a version of MD, contains a record each for the processor I- and D-banks beginning at the origins of the swap area and equal to the processor lengths.

Output: The library catalog is updated and a new processor absolute file created. The swap area extents (last required I-bank and D-bank addresses) necessary for processor loading are output to the terminal.



Relationship Between MDGENR and MDAS Memory Allocations

USAGE

THERE ARE TWO PROCEDURES FOR INVOKING MDGENR. THE FIRST IS USED WHEN PROCESSORS ARE TO BE ADDED OR CHANGED AND THE SECOND WHEN UTILITY PROCESSORS CONTAINED WITHIN THE SUBMONITOR ARE INVOLVED.

INVOKING MDGENR FOR PROCESSORS

```
!EDIT MDGENR MDGENP
AR T PRONAM:'.....',E      (SUPPLY PROCESSOR NAME AND
AR T PROENT:.....,E          ENTRY NAME, SEE RESTRICTIONS)
AQ
!SAS,N MDGENP
!GSLINK,WRITE,MORE,MAP MDGENP
AINCLUDE .....              (SUPPLY REQUIRED BLOCK DATA NAME)
AA:SEGMENT
AINCLUDE MDENDL
AEXIT
!MDGENP
!CHANGE MDGENP-PNC ACCESS:REPL
!UNLOAD
!DROP,EVERY MDGENP
```

INVOKING MDGENR FOR MONITOR CONTAINED UTILITY PROCESSORS

```
!EDIT MDGENR MDGENP
AR T PRONAM:'.....',E      (SUPPLY PROCESSOR NAME)
AR T PROENT:MDENDL,E
AR A A4,T-1:0,E
AQ
!SAS,N MDGENP
!GSLINK,WRITE MDGENP
!MDGENP
!CHANGE MDGENP-PNC ACCESS:REPL
!UNLOAD
!DROP,EVERY MDGENP
```

EXTERNAL REFERENCES

ECLOS\$ TO CLOSE FILES
ECTSO\$ TO OUTPUT TO TERMINAL
ELRSR\$ TO READ LOGICAL RECORDS
ELRSW\$ TO WRITE LOGICAL RECORDS
EOPENS TO OPEN FILES
EROOLS TO TERMINATE EXECUTION
ETRUNS TO TRUNCATE FILES
MDENDD TO LOCATE END OF ALLOCATED D-BANK
MDENDI TO LOCATE END OF ALLOCATED I-BANK
SUBSYSTEM PROCESSOR TO BE ADDED OR MODIFIED

RESTRICTIONS

WHEN PERFORMING THE DESIGNATED EDITS OF MDGENR THE NAME IN ' MARKS SHOULD BE BLANK FILLED TO THE RIGHT TO ASSURE SIX CHARACTERS. THE EDIT REVISIONS SHOULD ECHO THREE, TWO AND ONE LINES OF CODE IN TURN.

THE FOLLOWING SYMBOLS ARE USED IN MDGENR CODE AND THUS MUST NOT APPEAR AS THE PROCESSOR ENTRY POINT NAME.
AO-A15, ADRNAM, B1-B15, BLDASC, CRES, DOUT, ERMSG, ERROR, FILNAM, H1, H2, IOUT, LDOCB, LDUCB, M, MDENDD, MDENDI, MDFSTD, MDFSTI, MDGENR, MSG, MSG2, PTFM, PTPK, TABUFF, TBOCB, TBUCB, UTIL1, UTIL2, XH2, XM

MDGENR UPDATES THE CATALOG FILE MDTABL.MD ON WHICH EVER INFONET LIBRARY IT IS FOUND. IT WILL NOT WRITE INTO LIBS UNLESS THAT IS THE ORIGIN OF THE FILE.

DIAGNOSTICS

OUTPUT ERROR, ERROR ID IN A1

AN ERROR HAS OCCURRED DURING OUTPUT OF THE CATALOG OR PROCESSOR FILE. THE SYSTEM RETURN CODE IS CONTAINED IN REGISTER A1.

..... COPIED TO LOAD FILE EXTENT I D
THE DESIGNATED PROCESSOR HAS BEEN OUTPUT TO THE LIBRARY
THE LAST ALLOCATED ADDRESSES OF THE I- AND D-BANKS ARE SPECIFIED.

EXTERNAL STORAGE

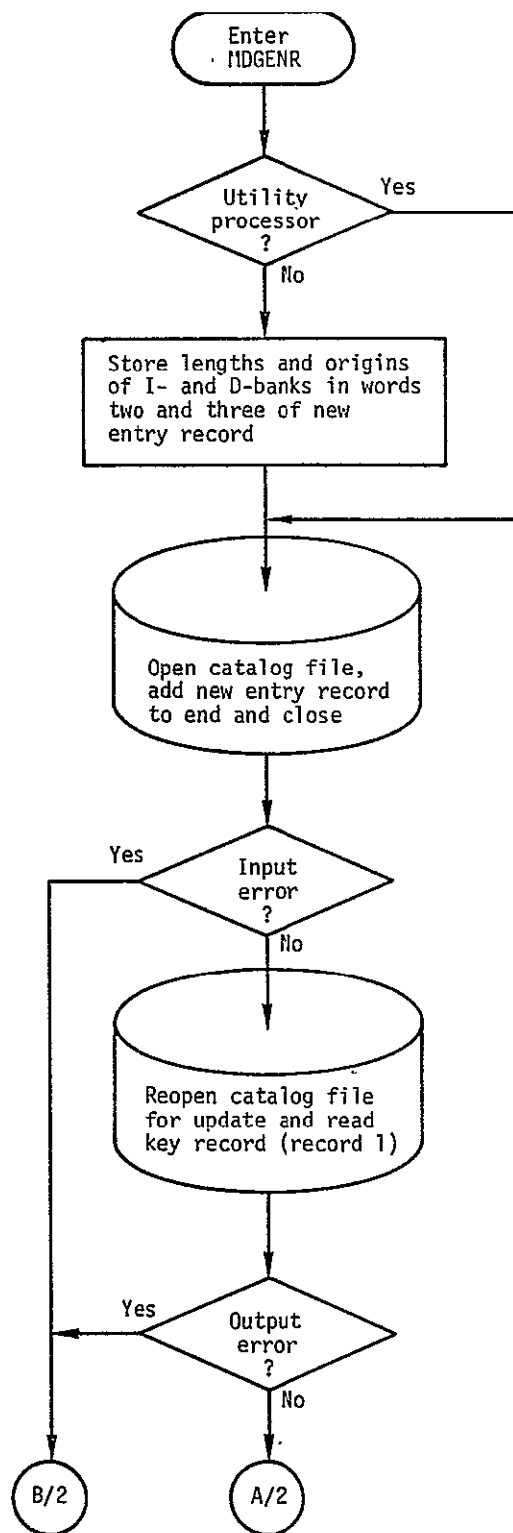
THE FILE MDTABL.MD IS MODIFIED TO REFLECT THE PROCESSOR FILE CREATED. A FILE NAMED AFTER THE PROCESSOR WITH VERSION MD IS OUTPUT.

BLANK COMMON

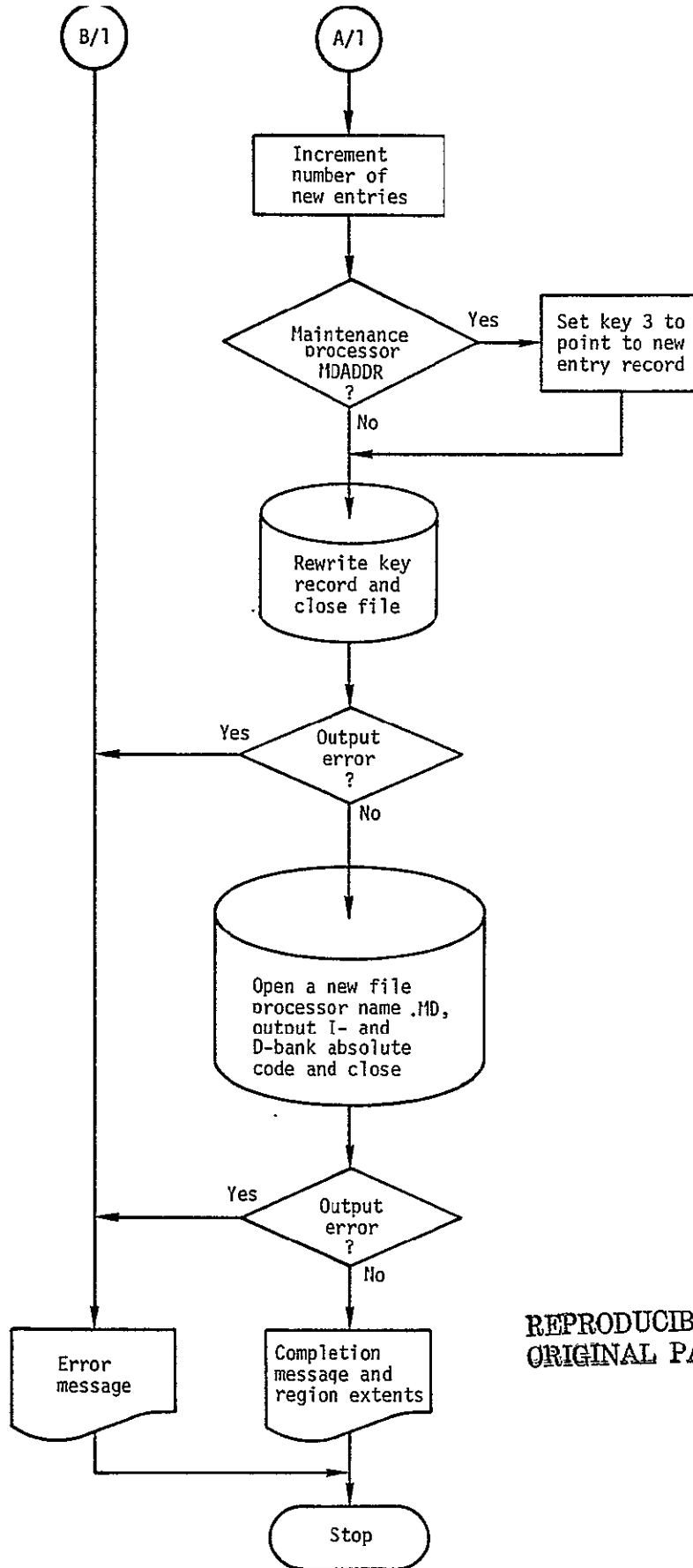
NONE

LOCAL COMMON

NONE



MDGENR - Library Maintenance Program



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDIMS - IMS Interface

MDIMS is intended to be the primary subroutine of the MDAS/IMS interface component. Since there is no interface with IMS in this prototype, this component does not currently exist. However, one array of data (GLOCON) which will eventually originate in IMS is emulated by MDIMS.

Method

Input: The input to the IMS interface component will consist of primarily the name and type of the desired data. Optionally, the subscript(s) specifying a displacement into the desired array may be input. The calling component may also provide the IMS interface component with a buffer for the data retrieved as well as an indicator of its length.

Processing: There are two entries into the IMS interface component. MDIMS is called to retrieve a particular element from IMS, move the data into a designated buffer, and create an SMT entry for this data element. MDIMS1 is called to retrieve a particular data element from IMS and create an SMT entry for it.

In the current much simplified version of MDIMS only one data element is "retrieved" from IMS. The global constants array (GLOCON) is stored here and moved to the SMT when either MDIMS or MDIMS1 is called. All other calls to the existing IMS interface cause an error message and return a status indicating that the requested element was not found in IMS.

Output: When the MDIMS entry is called a buffer is returned containing the data element requested. When either MDIMS or MDIMS1 are called the SMT directory and data area are updated via a call to MDPUT in order to enter the new data element. A status flag is returned when either entry is used. The status will indicate data successfully "retrieved" from IMS, data not found in IMS, or error returned from MDPUT.

USAGE

ENTRY MDIMS

CALL MDIMS (NAME, TYPE, IDIM, JDIM, MAX, BUFF, SIZE, STAT)

ARGMT	I/O	TYPE	DIM	DEFINITION
NAME	I	HOLL	I	NAME OF THE VARIABLE TO BE RETRIEVED
TYPE	I	I	I	TYPE OF THE VARIABLE TO BE RETRIEVED
IDIM	I	I	I	I-SUBSCRIPT USED TO DETERMINE FIRST WORD OF RETRIEVED VARIABLE TO BE TRANSFERRED TO BUFF.
JDIM	I	I	I	J-SUBSCRIPT USED TO DETERMINE FIRST WORD OF RETRIEVED VARIABLE TO BE TRANSFERRED TO BUFF.
MAX	I	I	I	MAXIMUM NUMBER OF WORDS TO BE TRANSFERRED INTO BUFF.
SIZE	O	I	I	NUMBER OF WORDS ACTUALLY TRANSFERRED INTO BUFF.
STAT	O	I	I	RETURN STATUS FLAG = -1 => VARIABLE NOT FOUND IN IMS (SGLOCON IS THE ONLY AVAILABLE IMS VARIABLE CURRENTLY) = 0 => O.K. OTHER => ERROR RETURNED FROM STORAGE MONITOR (MDPUT). VALUE IS 1 LESS THAN MDPUT'S STATUS.

ENTRY MDIMS1

CALL MDIMS1 (NAME, TYPE, STAT)

NAME	I/O	TYPE	DIM	DEFINITION
NAME	I	HOLL	I	NAME OF THE VARIABLE TO BE RETRIEVED
TYPE	I	I	I	TYPE OF THE VARIABLE TO BE RETRIEVED
STAT	O	I	I	RETURN STATUS FLAG = -1 => VARIABLE NOT FOUND IN IMS (SGLOCON IS THE ONLY AVAILABLE IMS VARIABLE CURRENTLY) = 0 => O.K. OTHER => ERROR RETURNED FROM STORAGE MONITOR (MDPUT). VALUE IS 1 LESS THAN MDPUT'S STATUS.

MDPUT

DIAGNOSTICS

*** MDIMS CALLED FOR TYPE=
 INDICATES THAT AN ATTEMPT TO RETRIEVE DATA FROM THE NON-EXISTENT IMS INTERFACE HAS BEEN MADE

*** MDIMS CALLED-- NAME= TYPE=
 IDIM= JDIM= MAX=
 INDICATES THAT AN ATTEMPT TO RETRIEVE DATA FROM THE NON-EXISTENT IMS INTERFACE HAS BEEN MADE

REPRODUCIBLE. OF THE
 ORIGINAL PAGE IS POOR

EXTERNAL STORAGE
NONE

BLANK COMMON
NONE

LOCAL COMMON
NONE

MDLOGO - Access Control

MDLOGO is the routine which controls use of the system and provides a measure of security for user created files. It also creates new entries in the access files when a user "logs on" the system for the first time.

Method

Input: MDLOGO has no input.

Processing: Upon entry, the user is prompted for an access code. The access file (MDACCD) is next read into the working buffer. The first record contains the number of active users and the total number of available codes. The second record contains the file identifier (version), the key (index) to the information file (MDUNIF), and both parts of the access code for each user. In addition it has all unassigned file versions and keyes with space reserved for future access codes.

If the user has entered a code which matches one in the file, control is returned to the calling routine and system operation begins. If the code does not match any in the file, the user is asked if he is a new user. If he is not, an access code must be entered which matches a previously defined one. A maximum of three attempts is allowed for the matching of an access code. When three attempts have failed, execution is terminated and control returned to the INFONET operation system.

When a user "logs on" the system for the first time, he is given a two character code which is used as a file identifier. In addition, he is asked to provide his name and organization which are inserted into the keyed information file (MDUINF). The access file (MDACCD) is sorted alphabetically on the first portion of the access code. Control is now returned to the calling routine and actual MDAS execution begins.

Output: All output is contained in the intramonitor communications area of blank common and consists of: the number of active users, the current users file identifier, and both parts of his access code.

Record
Number

Contents

1 (2 words)

Word 1	Word 2
Number of Active codes	Maximum # of codes

2
(3*N words)

Word 1	Word 2	Word 3
1 Assigned record 2 # in INFO 3 File . . . M	2 Char. version codes (FLDATA) . . . M	6 character field data access code assigned by the user 6 character field data code consisting of an organization code (e.g., T = TRW, L = LEC, etc.) followed by the 5 digits of the users badge number . . . M
1	2	3
N	N	N

Access Code File (MDACCD.MD)

Record
Number

1 (4 words)

Words 1 and 2	Word 3	Word 4
Last Name (max. 12 characters)	Initials	Organization

2

Last Name	Initials	Organization
-----------	----------	--------------

·
·
·

·
·
·

N

--	--	--

User Information File (MDUINF.MD)

USAGE

ENTRY MDLOGO
CALL MDLOGO

EXTERNAL REFERENCES

MDGETC
MDPRMT
MDPUTC
SEARCH
SORT1

RESTRICTIONS

MDLOGO WILL NOT MAINTAIN FILE INTEGRITY IF TWO, OR MORE, NFW
USERS ATTEMPT TO LOG-ON THE SYSTEM SIMULTANEOUSLY.

DIAGNOSTICS

ACCESS CODE OF IS NOT UNIQUE---TRY AGAIN
A NEW USER HAS ENTERED AN ACCESS CODE OF WHICH ONE
PORTION OF THE CODE ALREADY EXISTS
ACCESS TABLE IS FULL--SOMEONE MUST BE DELETED BEFORE ANY NEW
USERS MAY COME ON THE SYSTEM.
ALL AVAILABLE SLOTS FOR ACCESS CODES ARE BEING USED,
EITHER DELETE A USER OR INCREASE THE SIZE OF THE
AVAILABLE NUMBER OF ACCESS CODES
ERROR WHEN ATTEMPTING TO READ OR WRITE ACCESS FILE.
AN ERROR OCCURED IN ACCESSING MDACCD--NEED TO VERIFY
THE VALIDITY OF THE FILE
*I/O ERROR WHILE PROMPTING FOR BOOKKEEPING INFORMATION
PLEASE NOTIFY MDAS PROGRAMMING PERSONNEL
WHILE QUESTIONING FOR A NEW USER'S ORGANIZATION OR
NAME A PROMPTING ERROR OCCURED. THE USER IS ALLOWED
ON THE SYSTEM AND HIS VERSION IS CREATED; HOWEVER,
THERE IS NO ENTRY IN MDUINF FOR THIS USER.
**SYNTAX ERROR-ACCESS CODE HAS FORM CCCCCC.ABBBBB .
TYPE IN ? FOR A FULL EXPLANATION OF FIELDS
THE USER HAS MADE A SYNTAX ERROR WHEN ENTERING HIS
ACCESS CODE. ENTERING A ? GIVES A FULL EXPLANATION
OF THE NECESSARY SYNTAX
**SYNTAX ERROR ON INPUTTING NAME
AN ERROR OCCURED IN THE SYNTAX OF THE USER'S NAME.

EXTERNAL STORAGE

MDACCD

ACCESS FILE CONTAINING FILE VERSIONS
KEYS TO MDUINF, AND ACCESS CODES
INFORMATION FILE CONTAINING EACH
CURRENT USER'S NAME AND ORGANIZATION

MDUINF

BLANK COMMON

VARB I/O

ACCCDE 0
BDGNMB 0
NENTR 0

COMMON / MDCODE /

VARB I/O

COMMA I

EOL I

NAME I

QSTION I

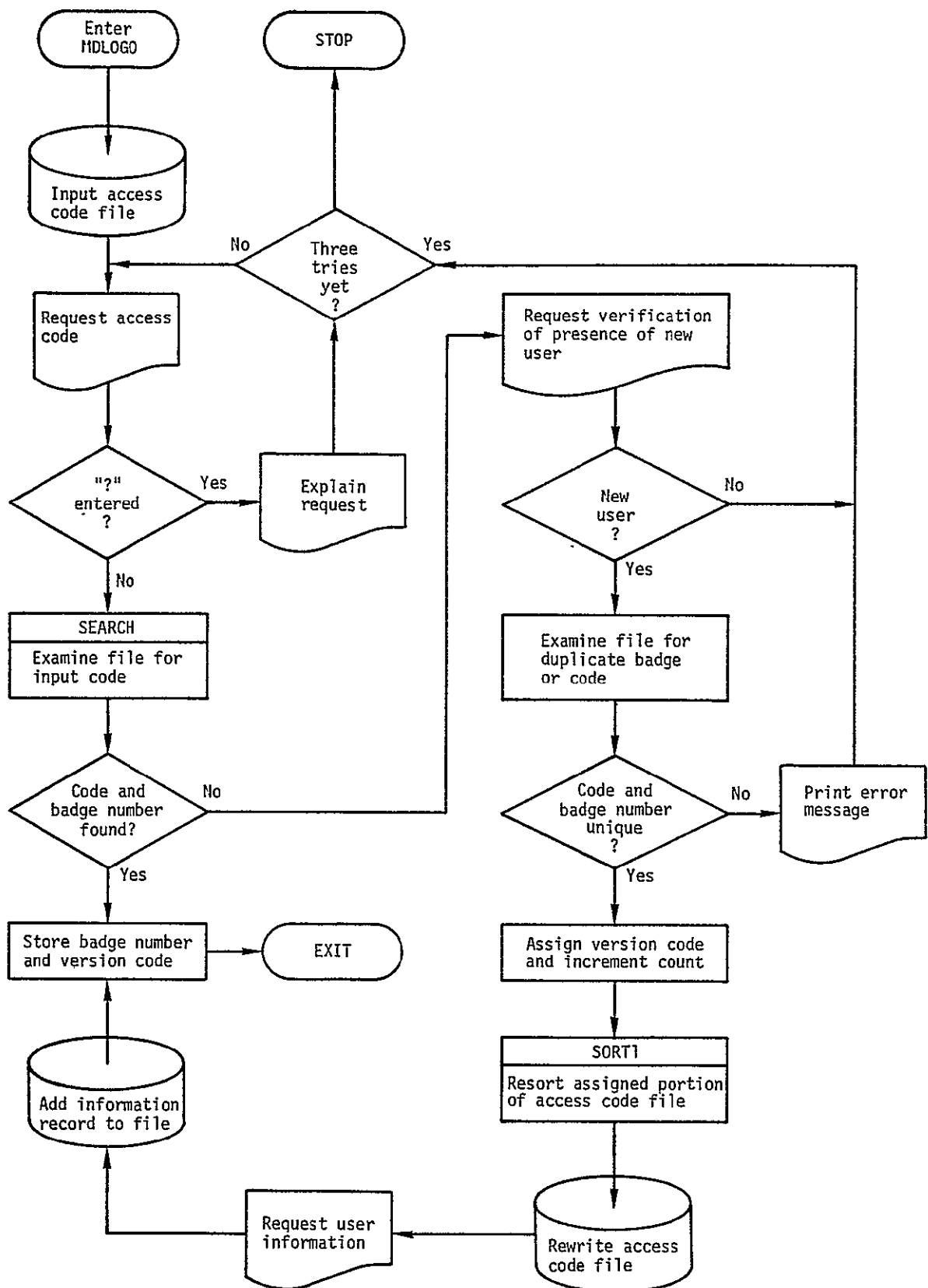
COMMON / MDBUFF /

VARB I/O

WB I

LOCAL COMMON

NONE



MDLOGO Functional Flow Diagram

MDELAC - User Accounting Files Maintenance Program

Purpose

Users of MDAS are uniquely identified by access code and badge number (see MDLOGO). Two accounting files are maintained which contain all necessary information regarding the access process. MDELAC is an auxiliary program which facilitates the maintenance of the files.

Method

Input: The operation to be performed by MDELAC, initialize files, delete user codes, or list the files, is input following prompts from the program. Specification of codes to delete is prompted following entry of the deletion mode.

Processing

The purpose of the initialization option is to purge the access code file such that only the MDAS maintenance code remains active. To accomplish this objective a file with the name MDACCD.MD of the following structure and content is created:

Record 1	Word 1	Word 2	
	1	163	

- Number of active codes and maximum number of codes.

1	bb	UPDATE	CC2531
2	PE		
3	PF		
4	PG		

21	PY		
22	PZ		
23	QE		
24	QZ		

161	WQ		
162	WR		
163	WS		

In addition, keyed record number one of the keyed file MDUINF.MD is filled with the following text:

Word 1	Word 2	Word 3	Word 4
MAINTE	NANCE	blank	TRW

The user code deletion option accesses the access code file MDACCD.MD and, under user control, deletes the requested codes from the list active access codes. A new file containing the remaining valid code is output.

Output: Except for the list option, MDELAC outputs the accounting files MDACCD.MD and MDUINF.MD.

USAGE

ENTRY MDELAC
EXECUTE THE GPS COMMAND !MDELAC

THE INPUTS TO MDELAC ARE AS FOLLOWS

AINIT CAUSES THE ACCOUNTING FILES TO BE PURGED ACCEPT FOR
THE MDAS MAINTENANCE ACCESS CODE
ADEL ENTERS A MODE OF DELETING INDIVIDUAL USER ACCFSS
CODES VIA THE FOLLOWING INPUTS
CCCCC CODE TO BE DELETED
ABBBBB BADGE NUMBER OF USER
#QUIT EXIT CODE DELETION MODE
ALIST DISPLAY THE ENTIRE ACCOUNTING DATA CONTENTS
AQUIT TERMINATE EXECUTION OF MDELAC

EXTERNAL REFERENCES

MDGETC, MDPUTC, SEARCH, SORT1

DIAGNOSTICS

USER/..... NOT IN SYSTEM
THE ACCESS CODE/BADGE NUMBER IS NOT AMONG THE ACTIVE
USER CODES.
ERROR IN STATUS =
MDGETC OR MDPUTC RETURNED THE INDICATED STATUS. REFERR
TO THE APPROPRIATE DOCUMENTATION FOR EXPLANATION.

EXTERNAL STORAGE

MDELAC INPUTS, MODIFIES AND OUTPUTS THE ACCESS CODE FILE
MDACCD.MD AND THE USER IDENTIFICATION FILE MDUINF.MD. UNIT 1
IS TEMPORARILY EQUATED TO MDUINF.MD.

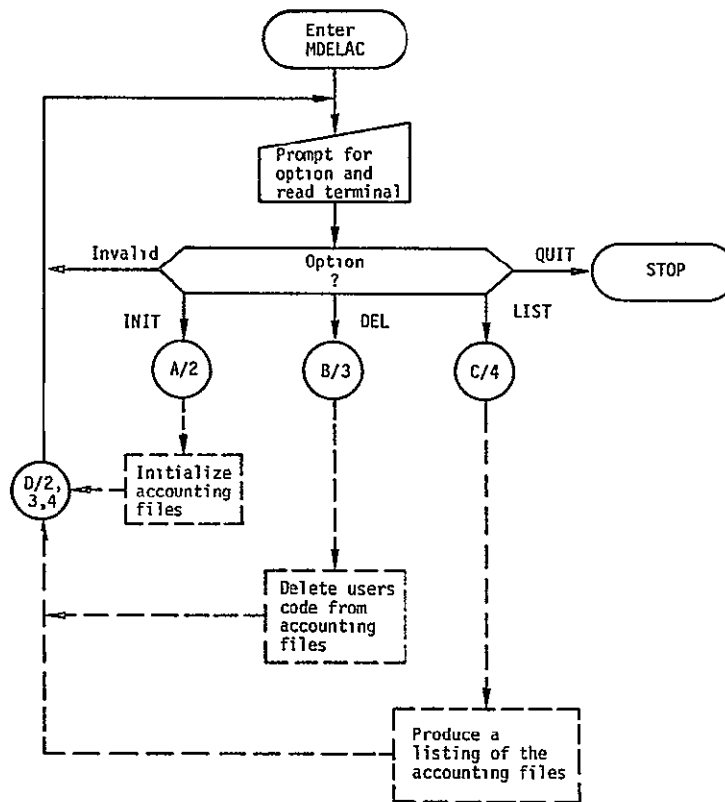
BLANK COMMON

NONE

LOCAL COMMON

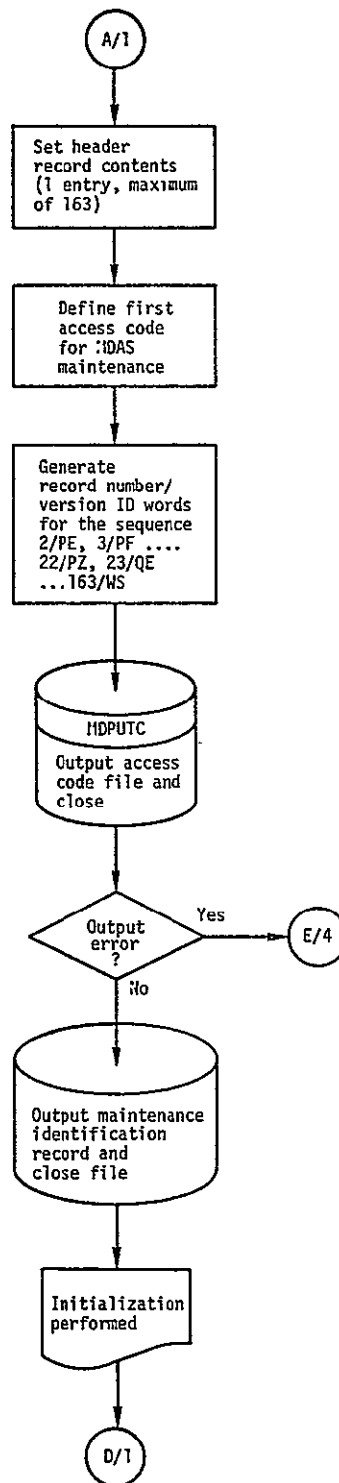
NONE

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



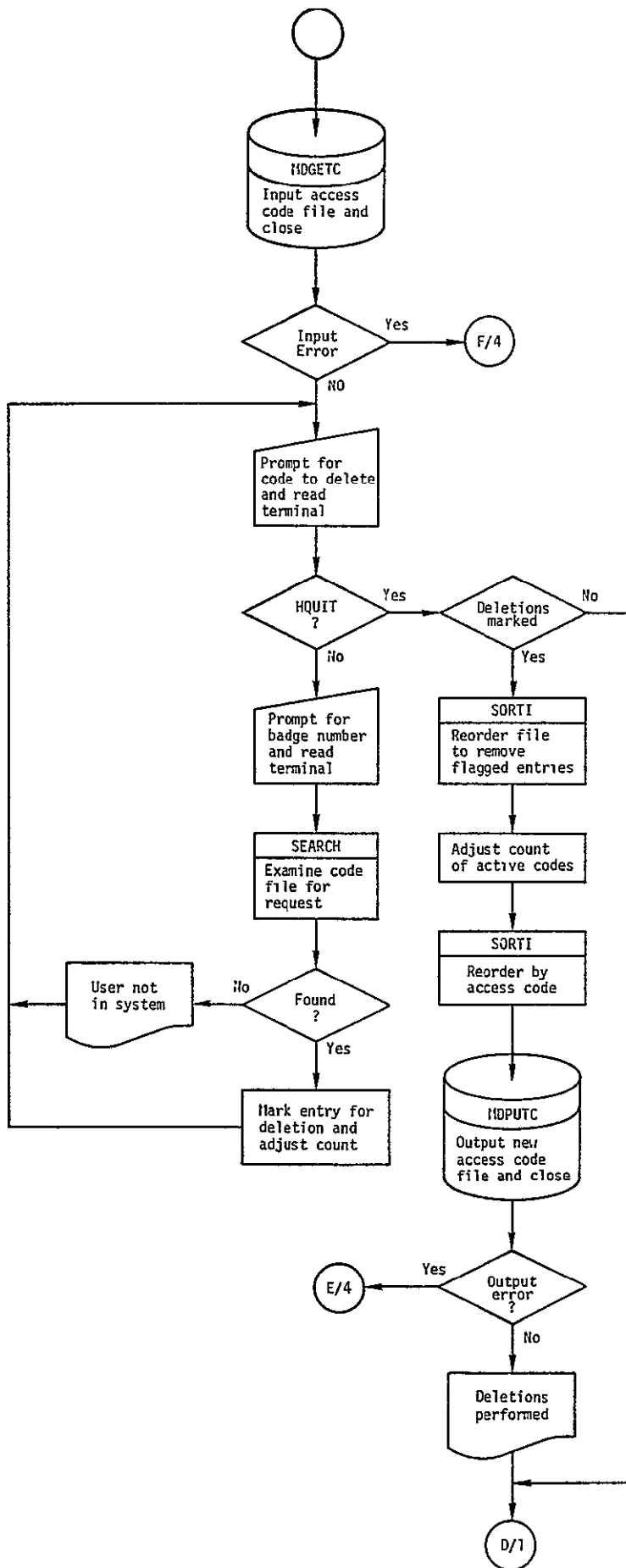
MDELAC Functional Flow Diagram

11.2-4

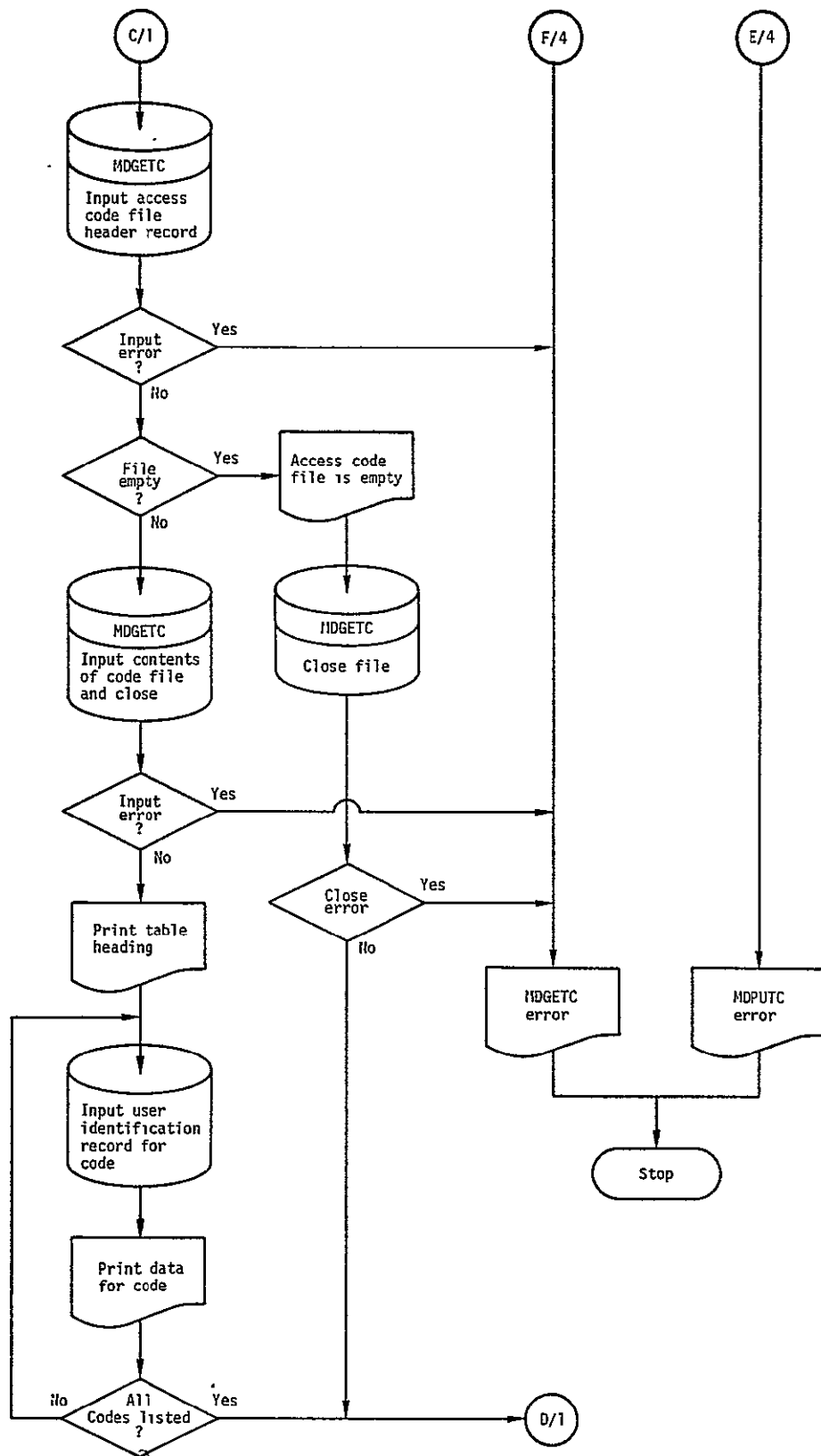


HDELAC - r INIT Logic Functional Flow Diagram

11.2-5



REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR



MDLAC -1 LIST Logic Functional Flow Diagram

Appendix A

Cross Reference of all Monitor Subroutines

Routine Referencing Routines

MDADDR								
MDALCT	MDSMON.							
MDALOC	MDALCT.	MDSMON.						
MDALST	MDCNT .	MDCONT.	MDLIST.					
MDBCDI	MDCDAT.							
MDBCI2	MDSCAN.							
MDBULD	DCTMOD.	MDADDR.						
MDCDAT	MDSCAN.							
MDCMNT	MDCMT .							
MDCMT	MDSMON.							
MDCMTG	MDSMON.							
MDCMTL	MDCMT .	MDLIST.	MDSMON.					
MDCMTS	MDCMT .	MDCMTG.	MDSMON.					
MDCMTV	MDSMON.							
MDCNT	MDSMON.							
MDCNTA	MDCONT.							
MDCNTE	MDCONT.							
MDCNTM	MDCMNT.							
MDCNTS	MDALOC.	MDCNT .	MDCONT.					
MDCONT	MDCNT .							
MDCONV	MDELET.	MDGETC.	MDPRMT.	MDPUTC.	MDSMON.			
MDCTPK	MDALOC.	MDBULD.	MDCNT .					
MDDEFN	MDCNT .							
MDEDCN	MDBULD.	MDSMON.						
MDEDIT	MDCONT.							
MDELET	MDALCT.	MDALOC.	MDCMNT.	MDPUT .	MDQUIT.	MDUTIL.		
MDENDD								
MDENDI								
MDENDL								
MDENTR	MDALCT.	MDALOC.	MDPUT .					
MDFIND	MDALCT.	MDALOC.	MDCMTG.	MDELET.	MDGET .	MDPUT .	MDSMON.	SMPRTP.
MDGET	MDCMNT.	MDCMT .	MDCNT .	MDCNTE.	MDSMON.	MDUTIL.		
MDGETC	MDBULD.	MDCMNT.	MDCNT .	MDELAC.	MDLOGO.	MDSMON.	MDSMTW.	

REPRODUCIBILITY OF THE
ORIGINAL PAGE IS POOR

MDIRCD									
MDIMS	MDCNT .	MDCNTE.							
MDIMS1	MDALOC.								
MDLIST	MDBULD.	MDCNT .	MDUTIL.						
MDLKUP	MDALCT.	MDALOC.	MDALST.	MDCNT .	MDCONT.				
MDLOGO	MDADDR.	MDSMON.							
MDLSTH	MDALST.	MDCMTL.	MDLIST.						
MDLSTI	MDALST.	MDCMTL.	MDLIST.						
MDLSTO	MDALST.	MDCMTL.	MDLIST.						
MDLSTR	MDALST.	MDCMTL.	MDLIST.						
MDMERG	MDSMON.								
MDPACK	MDENTP.	MDQUIT.	MDROLL.	MDTOC .					
MDPCK	MDCDAT.	MDSCAN.							
MDPRMH	MDALST.								
MDPRMI	MDALST.								
MDPRMR	MDALST.								
MDPRMT	LPSVFC.	LSV .	MDALST.	MDBULD.	MDCMT .	MDCNT .	MDLOGO.	MDPRMH.	MDPRMI
	MDQUIT.	MDSMON.	MDSMTW.	MDPRMR.					
MDPUT	MDALOC.	MDCMNT.	MDCMT .	MDCNT .	MDIMS .				
MDPUTC	MDALOC.	MDBULO.	MDELAC.	MDLOGO.	MDSMTW.				
MDQUIT	MDSMON.								
MDRADI	MDGET .								
MDRADO	MDPUT .								
MDROLL	MDALOC.								
MDSCAN	MDPRMT.								
MDSMON	MDAS .								
MDSMTR	MDSMON.								
MDSMTW	MDQUIT.	MDSMON.							
MDSPEC	MDALOC.	MDCNT .	MDCNTS.	MDLIST.					
MDSPLT	MDBULD.	MDCMT .	MDCNT .	MDSMON.	MDSMTW.				
MDSQZB	MDBCDT.	MDCDAT.							
MDTOC	MDSMON.								
MDUTIL	MDSMON.								
MDVCMD	MDCMT .	MDSMON.							
OBEY	DCTMON.	MDADDR.	MDELAC.	MDELET.	MDLOGO.	MDQUIT.	MDSMON.		
SEARCH	DCTMON.	MDADDR.	MDBULD.	MDCMNT.	MDCNT .	MDELAC.	MDFIND.	MDLOGO.	MDSMON

SORT1	MDADDR.	MDCMT .	MDELAC.	MDFIND.	MDLOGO.	MDPACK.
SORT2						
UPDATE	MDLOGO.					

Appendix B

Common Blocks

COMMON /MDCODE/ ALLOCATION

VARB	TYPE	DIM	LOC	RELADD	DEFINITION
NAME	I	1	C0DE(1)	0000	FIELD DESIGNATOR INDICATING AN ALPHNUMERIC NAME (=1)
REAL	I	1	C0DE(2)	0001	FIELD DESIGNATOR INDICATING A REAL NUMBER (=2)
INTEG	I	1	C0DE(3)	0002	FIELD DESIGNATOR INDICATING AN INTEGER NUMBER (=3)
DBLE	I	1	C0DE(4)	0003	FIELD DESIGNATOR INDICATING A DOUBLE PRECISION NUMBER (=4)
EOS	I	1	C0DE(5)	0004	FIELD DESIGNATOR INDICATING THE END OF THE INPUT STATEMENT (=5)
HOLL	I	1	C0DE(6)	0005	FIELD DESIGNATOR INDICATING A HOLLERITH VALUE (=6)
OCTAL	I	1	C0DE(7)	0006	FIELD DESIGNATOR INDICATING AN OCTAL NUMBER (=7)
LPAR	I	1	C0DE(8)	0007	FIELD DESIGNATOR INDICATING A LEFT PARENTHESIS (=8)
RPAR	I	1	C0DE(9)	0010	FIELD DESIGNATOR INDICATING A RIGHT PARENTHESIS (=9)
DOLLAR	I	1	C0DE(10)	0011	FIELD DESIGNATOR INDICATING A DOLLAR SIGN, \$, (=10)
AT	I	1	C0DE(11)	0012	FIELD DESIGNATOR INDICATING AN AT SIGN, @, (=11)
PERCNT	I	1	C0DE(12)	0013	FIELD DESIGNATOR INDICATING A PER CENT SIGN, %, (=12)
COLON	I	1	C0DE(13)	0014	FIELD DESIGNATOR INDICATING A COLON, :, (=13)
APOSTR	I	1	C0DE(14)	0015	FIELD DESIGNATOR INDICATING AN APOSTROPHE, ', (=14)
EQUALS	I	1	C0DE(15)	0016	FIELD DESIGNATOR INDICATING AN EQUALS SIGN, =, (=15)
MINUS	I	1	C0DE(16)	0017	FIELD DESIGNATOR INDICATING A MINUS SIGN OR A HYPHEN, -, (=16)
COMMA	I	1	C0DE(17)	0020	FIELD DESIGNATOR INDICATING A COMMA, (=17)
UPARRW	I	1	C0DE(18)	0021	FIELD DESIGNATOR INDICATING AN UP-ARROW, ↑, (=18)
BCKSLH	I	1	C0DE(19)	0022	FIELD DESIGNATOR INDICATING A BACK-SLASH, \, (=19)
QUESMK	I	1	C0DE(20)	0023	FIELD DESIGNATOR INDICATING A QUESTION MARK, ?, (=20)

VARB	TYPE	DIM	LOC	RELADD	DEFINITION
PLUS	I	1	CODE(21)	0024	FIELD DESIGNATOR INDICATING A PLUS SIGN, +, (=21)
ASTRSK	I	1	CODE(22)	0025	FIELD DESIGNATOR INDICATING AN ASTERISK, *, (=22)
LBSIGN	I	1	CODE(23)	0026	FIELD DESIGNATOR INDICATING A POUND SIGN, #, (=23)
SLASH	I	1	CODE(24)	0027	FIELD DESIGNATOR INDICATING A SLASH, /, (=24)
			CODE(25) TO CODE(32) NOT CURRENTLY USED		
SUBS	I	1	CODE(33)	0040	FIELD DESIGNATOR INDICATING A SUBSCRIPT FIELD (=33)
			CODE(34) TO CODE(42) NOT CURRENTLY USED		
REPEAT	I	1	CODE(43)	0052	FIELD DESIGNATOR INDICATING A REPEAT GROUP (=43)

COMMON /MDBUFF/ ALLOCATION

VARB	TYPE	DIM	RELADD	DEFINITION
MDLEN	1	1	0000	LENGTH (IN WORDS) OF THE WORKING BUFFER (WBUF)
BDATA	1	1	0001	SUBSCRIPT (ONE ORIGIN) FROM BEGINNING OF WORKING BUFFER (WBUF) TO BEGINNING OF ITS DATA AREA (I.E. PORTION OF WBUF WHICH GROWS UP FROM THE BOTTOM)
DSIZE	1	1	0002	TOTAL NUMBER OF WORDS OF THE WORKING BUFFER (WBUF) WHICH ARE CURRENTLY IN USE. THE WORKING BUFFER IS DIVIDED INTO TWO AREAS OF DATA -- ONE AT THE TOP AND ONE AT THE BOTTOM. DSIZE IS THE TOTAL SIZE OF THESE TWO AREAS.
WBUF	1	MDLEN	0011	WORKING BUFFER OF MDLEN WORDS

WORDS 4 TO 9 OF /MDBUFF/ ARE NOT USED